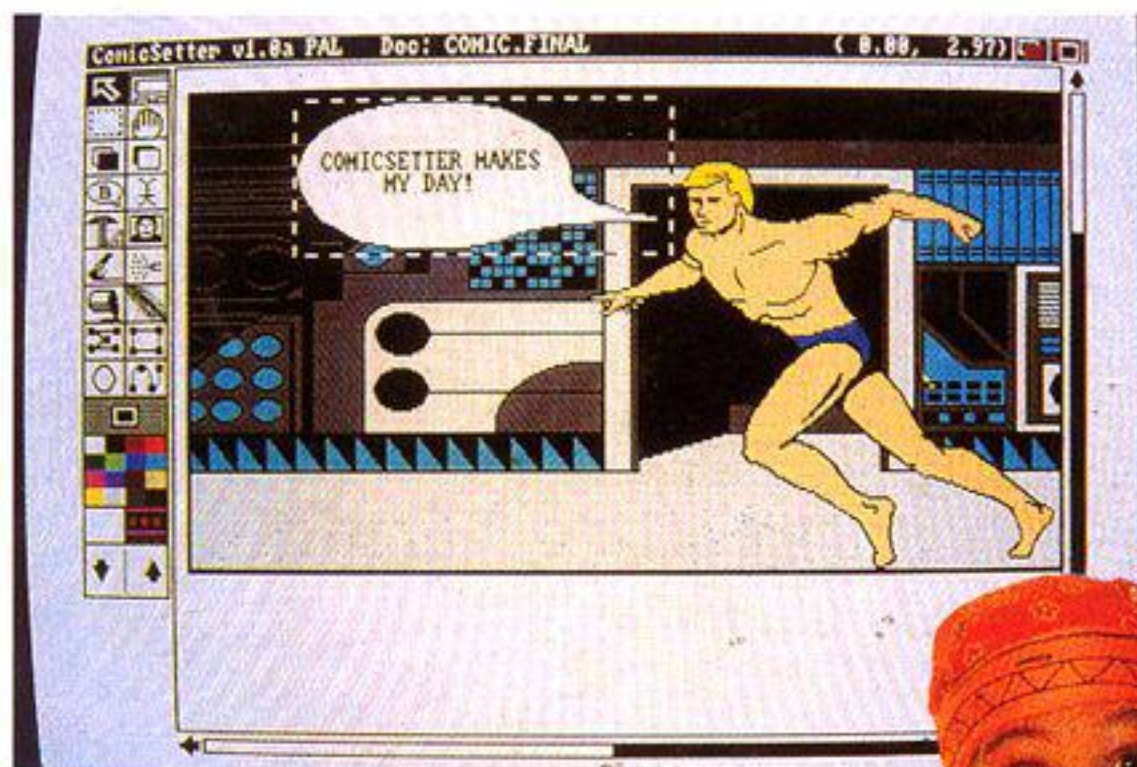
Commodore
COMPUTER CLUB
82 L. 6.000
La rivista degli utenti di sistemi Commodore

Anno XI - N. 82 - Marzo 1991 - Sped. Abb. Post. Gr III/70 - CR - Distr.: Parrini

PROVA
Tavoletta grafica contro Track-ball e mouse

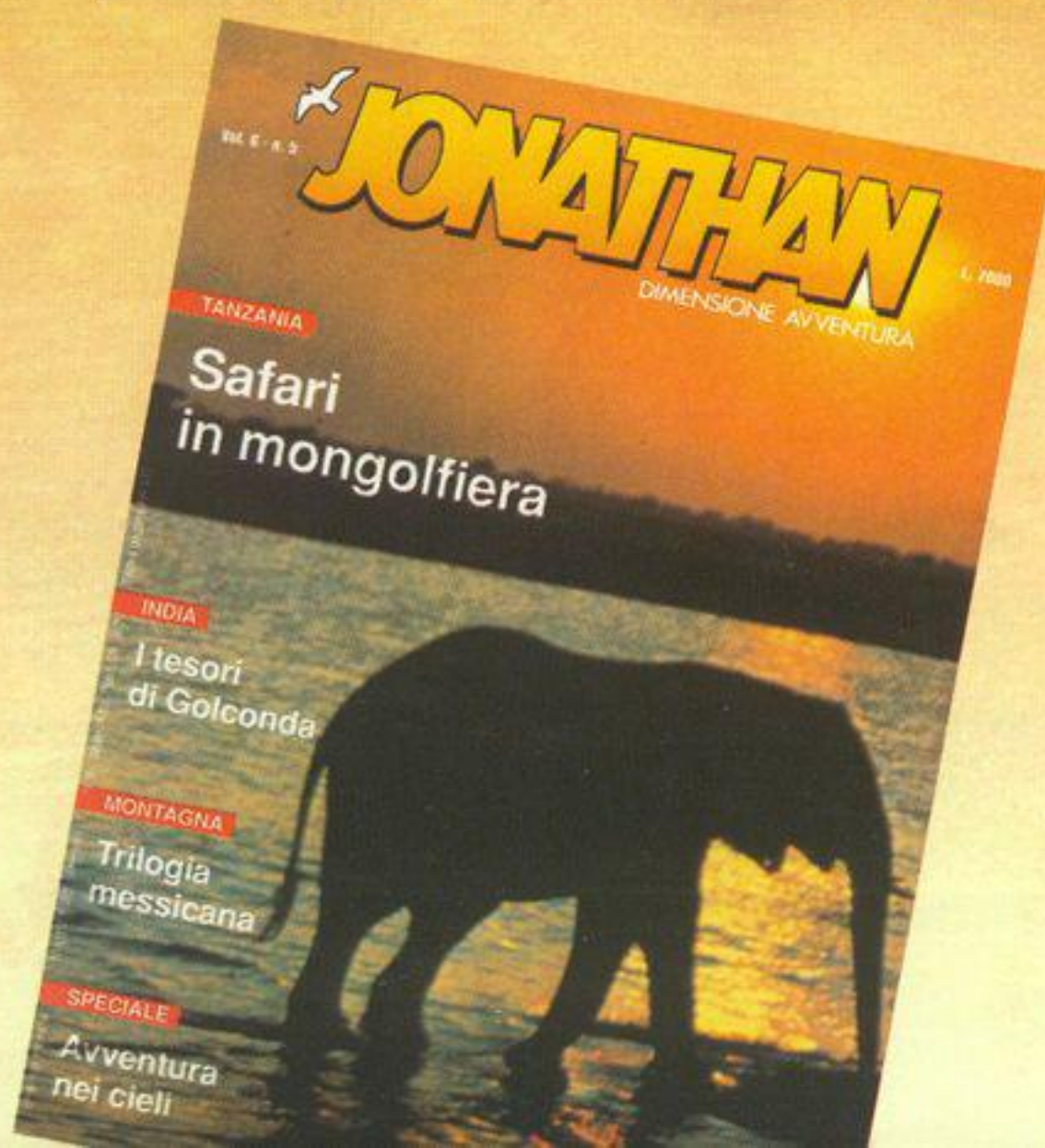
COMIC SETTER
Oggi invento Paperino

C 64/128
Come esplodere gli sprite

COMPILATORI
Quick Pascal MicroSoft

FANATIC ULTRA STYLE

AMIGA, finalmente "C"

Guida rapida a Pc Tools • Trasforma i files sequenziali in relativi • Ricordare le date con MS-DOS • Convertire testi da Amiga a Pc • Videogames per Amiga • Amiga Basic e le coordinate geografiche • Amigados • Enciclopedia di routines

Systems

Sommario

Amigames

Sono soltanto tre i videogames recensiti questo mese; ciò per venire incontro alle numerose richieste dei nostri lettori, che preferiscono un maggior numero di pagine dedicate ad articoli tecnici.

Spazio 64

Amiga

Mondo Dos

Amiga + Ms - Dos

COMMODORE COMPUTER CLUB

Direttore: Alessandro de Simone
Coordinatore: Marco Miotti

Redazione / Collaboratori:
Davide Ardizzone - Claudio Baiocchi
Luigi Callegari - Umbero Colapicchioni
Donato De Luca - Carlo d'Ippolito
Valerio Ferri - Michele Maggi
Giancarlo Mariani - Domenico Pavone
Armando Sforzi - Dario Pistella
Fabio Sorgato- Valentino Spataro
Franco Rodella- Stefano Simonelli
Luca Viola

Direzione:
Via Mosè, 22 cap. 20090 OPERA (Mi)

Telefono 02 / 57.60.63.10
Fax 02 / 57.60.30.39
BBS 02 / 57.60.52.11

Pubblicità:
Leandro Nencioni (dir. vendite)
Via Mosè, 22 20090 Opera (Mi)
tel. 02 / 57.60.63.10

Emilia Romagna:
Spazio E
P.zza Roosvelt, 4 cap. 40123 Bologna
Tel. 051 / 23.69.79

Toscana, Marche, Umbria
Mercurio s.r.l. Via Rodari, 9
S. G.nni Valdarno (Ar)
Tel. 055 / 94.74.44

Lazio, Campania
Spazio Nuovo
Via P. Foscari, 70
cap. 00139 Roma
tel. 06 / 81.09.679

Abbonamenti: Liliana Spina
Arretrati e s/w: Lucia Dominoni

Tariffe: Prezzo per copia L. 6000
Abbonamento annuo (11 fascicoli) L. 60000
Estero: L. 100000 - Indirizzare versamenti a:
Systems Editoriale Srl c/c 37952207 oppure
inviare comune assegno bancario non
trasferibile e barrato due volte a:
Systems Editoriale Srl (servizio arretrati)
Via Mosè, 22
cap. 20090 OPERA (Mi)

Composizione: Systems Editoriale
La Litografica Srl Busto Arsizio (Va)

Registrazioni: Tribunale di Milano
n. 370 del 2/10/82

Direttore Responsabile: Michele Di Pisa

Spedizioni in abbonamento postale
gruppo III. Pubblicita' inferiore al 70%

Distributore: Parrini - Milano

Periodici Systems: Banca Oggi - Commodore Club (disco) - Commodore Computer Club - Commodore Computer Club (disco, produzione tedesca) - Computer - Computer disco - Electronic Mass Media Age - Energy Manager - Hospital Management - Jonathan - Nursing '90 - PC Programm (disco) - Personal Computer - Security - Software Club (cassetta ed. italiana) - TuttoGatto - Videoteca VR Videoregistrare

Bello senz'anima

Le software house hanno preso il vizio delle hardware house. Vale a dire che provvedono allo sviluppo di un progresso quasi fine a se stesso, senza tener conto se, effettivamente, tali sofisticazioni sono realmente richieste da tutti gli utenti interessati, oppure no.

Prendiamo, a caso, due s/w appartenenti di altrettanti "tipi" di programmi; un word processor (che facilita, e di molto, la stesura di testi) ed uno di utilità (come può essere un'interfaccia amichevole per la gestione dei files su disco).

Se compariamo tra loro, ad esempio, due "storiche" versioni di Word Star (ciè la 3 e l'ultima) non possiamo nasconderci che la precedente era piuttosto "grezza" e poteva reggere il confronto, addirittura, con altri w/p che giravano su computer di più modeste caratteristiche, tra cui il C/64.

L'attuale versione di W/S consente sofisticazioni prima impensabili (basta pensare alla preview, che permette di simulare, su video, l'aspetto finale del documento) di cui, *forse*, si sentiva la carenza nelle precedenti release.

I manuali acclusi alla confezione, però, hanno incrementato la propria consistenza in diretta proporzione con le potenzialità del software. Allo stesso modo, una precedente versione di utilità (e citiamo, ancora a caso, il famoso PC Tools, giunto alla sesta edizione) presentava, magari, alcune carenze, eliminate nel corso delle successive versioni. Ma anche in questo caso i manuali (per non parlar del prezzo finale al pubblico) sono cresciuti in proporzione, allo scopo di offrire tutte le spiegazioni del caso all'utente.

Mettiamoci, ora, nei panni di quest'ultimo, che vorrebbe entrare in possesso, possibilmente, di s/w semplice, di immediata comprensione e di basso costo.

Egli è ben felice di essere il possessore di *"uno dei più completi e potenti pacchetti mai sviluppati finora"* ma, ne siamo sicuri, avrebbe preferito la possibilità di impadronirsi del programma con maggiore velocità (e, se possibile, risparmiando qualche liretta) pur se a discapito di qualche potenzialità che, forse, non sfrutterà mai, ma che ha partecipato a far lievitare sia il prezzo, sia la corposità del manuale, sia la necessità di studiare menu e sub-menu che contengono informazioni ad essa relative.

Qual è, dunque, il "vizio" che rimproveriamo alle s/w house? Ci riferiamo all'errata convinzione che l'utente finale sia identico, per usi e costumi, agli stessi progettisti delle s/w; si ritiene, erroneamente, che egli passi il suo tempo alla perenne ricerca di perfezionismi impensabili e di sofisticherie utili in casi improbabili.

Agnelli non si sognerebbe mai di vendere solo la Ferrari Testarossa, dal momento che rappresenta il top tecnologico oggi raggiungibile nel campo automobilistico. In vendita, partendo dalla Fiat 126, troviamo infatti in listino una vasta categoria di autovetture, adatte ad altrettante categorie di utenti ed in grado di risolvere vari problemi, tutti diversificati tra loro.

E allora ci chiediamo: perchè mai offrire sempre e solo un unico programma, pur se questo rappresenta il meglio che la s/w house ha mai prodotto? Non potrebbero venire offerti vari "modelli" dello stesso s/w, pur se perfettamente compatibili tra loro, ma uno meno sofisticato (e, soprattutto, costoso) dell'altro? L'utente, dopo aver compreso le potenzialità della versione più semplice, potrebbe decidere, se necessario, di passare a quella superiore spendendo una cifra ridotta e, soprattutto, sicuro che il "passaggio" non lo obbligherà ad imparare tutto daccapo, ma soltanto i comandi che risultavano assenti nella versione meno potente. Se, invece, i limiti della versione economica sono considerati accettabili, egli continuerà ad usarla così com'è, felice di essere entrato in possesso di un s/w a misura di utente; ma sempre pronto, se le esigenze dovessero aumentare, a prendere in considerazione l'acquisto dell'upgrade.

E', pertanto, con spirito... polemico che, nella parte interna di questo fascicolo, troverete una "recensione" relativa ad una vecchia (ma altrettanto valida) versione di un programma di utilità, facilissimo da apprendere, diffusissimo tra gli appassionati, ma forse sconosciuto ai neo - utenti che da poco si sono procurati un meraviglioso computer Ms - Dos.

Già, ci sono anche loro; provengono da un'area informatica di modeste pretese; i programmi cui erano abituati erano corredati da un fascicoletto di semplice e rapida comprensione. Altri utenti, invece, sono ancora indecisi a compiere il gran passo forse perchè spaventati dalla corposità dei manuali e dalla difficoltà di comprenderli.

E, non ultimo, dal prezzo di vendita.

Alessandro de Simone

LA VOSTRA POSTA

(a cura di A. de Simone)

Voglia di scrivere

Vorrei scrivere un articolo riguardante i vari metodi di "impacchettamento". L'articolo riguarderebbe sia l'Amiga (ARC, ZOO, LHARC, LHARCA, PKAZIP, WARP e LHWARP) che il C/64 (ARC e PKZIP) ed amplierei il discorso anche all'MS - Dos.
(Ascanio Orlandini)

Tutti le notizie che possono interessare i programmatori (ed i telemaniaci in particolare) sono benvenute. Piuttosto interessante è l'idea (suggerita nella lettera dal nostro collaboratore) di scrivere l'articolo non solo per un utilizzo telematico dei compattatori, ma anche per impiegarli nell'esecuzione di back-up di file dati prodotti da WP, fogli elettronici, ed altro mediante impiego di files script (cioè files batch) che automaticamente consentirebbero di includere nell'archivio i files nuovi o più recenti rispetto a quelli già archiviati, con un notevole risparmio di spazio (per testi ASCII si risparmia un buon 60 / 70%). E' indispensabile, però, un colloquio "a viva voce" (per telefono) allo scopo di concordare la stesura dell'articolo.

❑ ❒

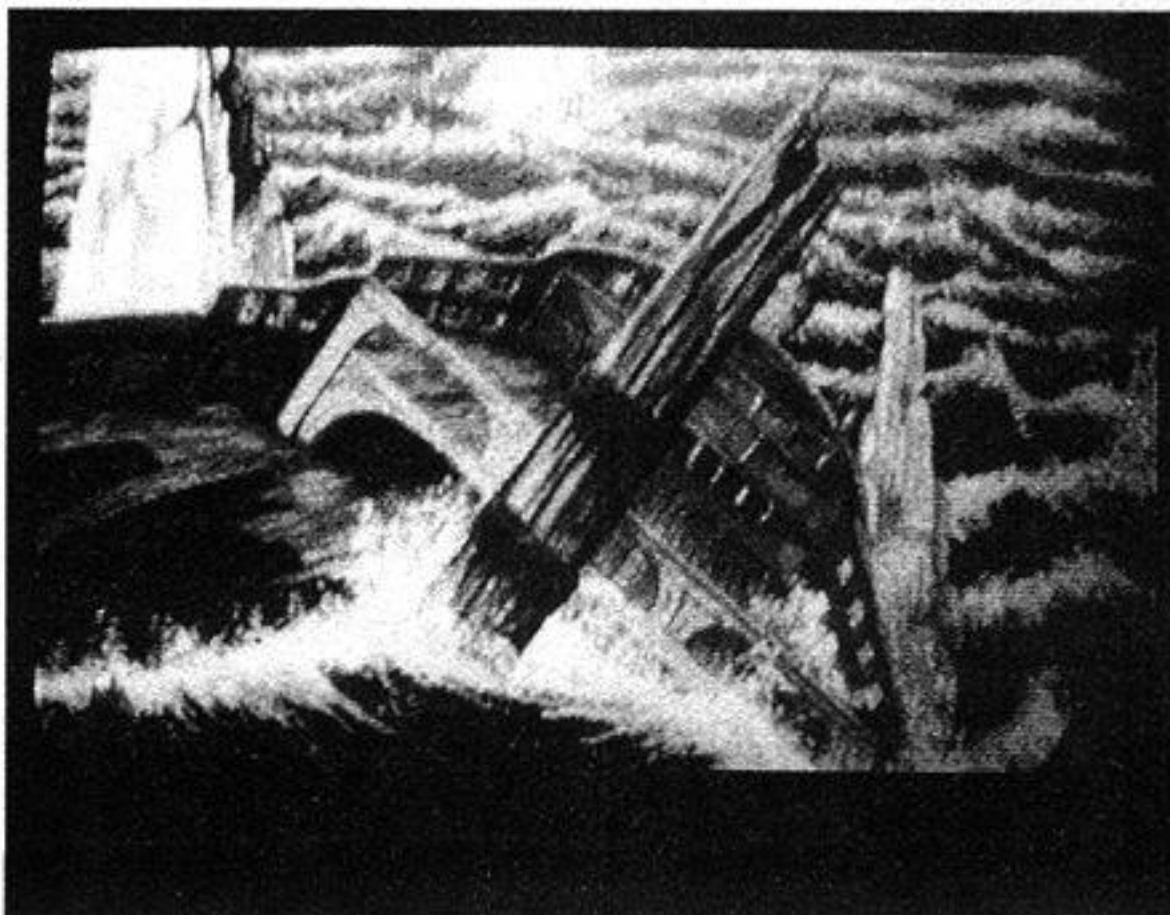
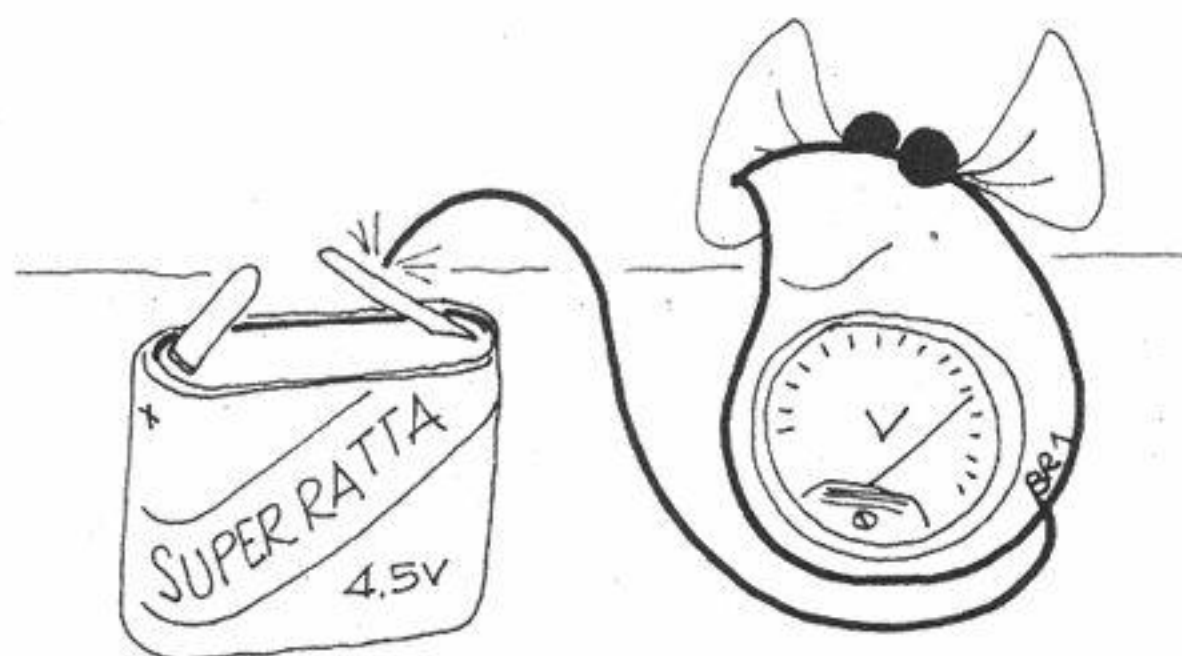
Schede grafiche Ms - Dos

L'articolo sulle schede grafiche apparso sul n. 80 di C.C.C. mi ha chiarito alcuni concetti, ma ha fatto sorgere diversi dubbi. Come faccio a sapere se il computer Ms - Dos che ho intenzione di acquistare consentirà, in futuro, la sostituzione della scheda grafica? Come faccio a conoscere la reale espandibilità del sistema?
(Carmine Musso - Portici)

Mi dispiace che il testo dell'articolo citato, volutamente evasivo (proprio per mettere in guardia i lettori meno informati), faccia trascorrere notti insonni al nostro simpatico lettore. I dubbi sul computer da acquistare, però, si risolvono telefonando al servizio di assistenza della marca menzionata; oppure rivolgendosi ad un **serio** rivenditore che, magari consentendo l'osservazione dell'interno del computer in questione, sia in grado di dimostrare che la sostituzione della scheda grafica sia effettivamente possibile.

Rimane da chiedersi come mai le case costruttrici non offrano sempre su scheda (anzichè sulla piastra madre) gli accessori collegabili al computer; in caso di occorrenza, infatti, sarebbe sufficiente sfilare la vecchia scheda per sostituirla con quella nuova.

La risposta (potevate dubitarlo?) è legata al guadagno che è possibile realizzare. Costruire un'unica scheda sulla quale è montata la maggior parte dei componenti possibili, consente un risparmio considerevole rispetto all'alternativa di fabbricare schede madri dotate di connettori in cui inserire altre schede.

In alcuni casi (computer portatili, al giorno d'oggi quasi "palmari") integrare tutto su un'unica scheda è un'esigenza irrinunciabile. In altri casi, invece, se ne potrebbe fare a meno. Una precisa scelta **commerciale**, quindi, è responsabile di alcune incomprensibili (per gli utenti finali) situazioni.

E' probabile che, nonostante la scheda grafica faccia parte della scheda madre, sia disponibile un deviatore che, opportunamente settato, consenta l'esclusione totale della circuiteria video e la conseguente possibilità di aggiungere la scheda desiderata.

L'unico modo di saperlo, ripeto, è solo quello di rivolgersi ad un rivenditore che abbia interesse a trattare un (probabile) cliente con la massima serietà.

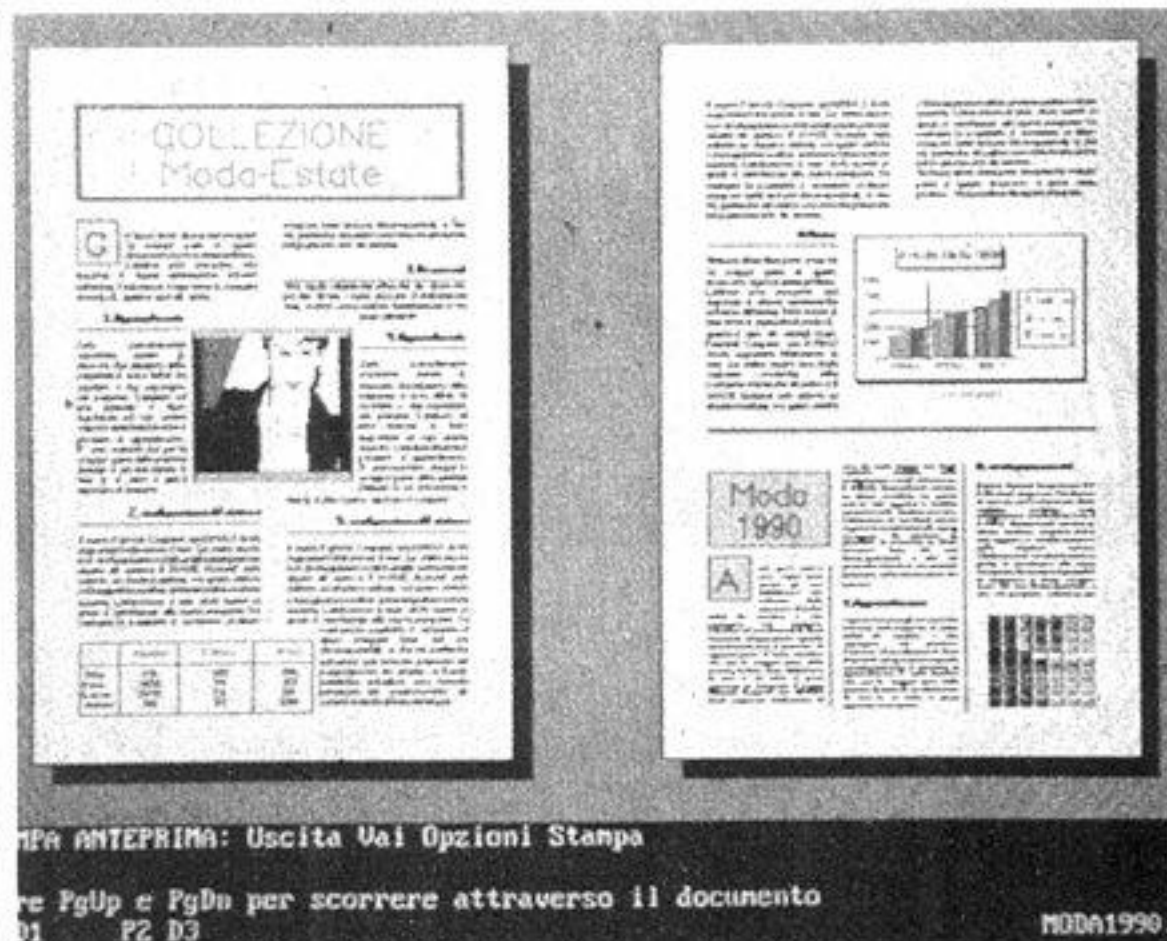
E di rivenditori di questo tipo, credetemi, ce ne sono parecchi.

Basta cercarli...

Da C/64 a QuickBasic

Con il vecchio C/64 potevo realizzare un Input molto comodo...

A$ = "Pippo": Input a$
Print A$

In altre parole, se al momento dell'Input, battevo il tasto Return senza digitare nulla, alla variabile A$ rimaneva associata la stringa precedente, cioè "Pippo". Se uso la stessa tecnica con il QuickBasic (installato sul mio computer Ms - Dos compatibile), alla stringa A$ viene invece associata una stringa nulla. Possibile che

Word per Window

Una altro testo in italiano, semplice da consultare, ha visto la luce presso la casa editrice **Apogeo.** Si tratta del volume **"Word per Window"** (di *Donna Sakson* e *Carol Taylor*) tradotto per la gioia degli utenti italiani. Il libro, prezzato a sole 48 mila lire, consta di 329 pagine suddivise in ben 22 capitoli.

Si parte dall'introduzione, che si occupa di informazioni generali sul potente pacchetto **Microsoft**, e si passa alla descrizione degli strumenti di base (schermo, tastiera, mouse); alle operazioni di gestione del testo (taglia, incolla, formattazione del testo); all'uso dei modelli predefiniti e personalizzabili; alle tecniche di editing; all'uso del colore, eccetera. Nulla, insomma, viene trascurato per mettere in evidenza, mediante semplici esempi, tutte le notevoli potenzialità del word processor marcato Microsoft che, in effetti, rappresenta anche un sistema di impaginazione particolarmente efficace.

Le numerosissime illustrazioni a corredo, riproducono le schermate che compaiono durante le fasi più importanti della gestione di un testo. Una particolare cura, prestata nel riportare caratteri tipografici differenti, consente al lettore di individuare facilmente la notizia che più interessa e, soprattutto, di seguire le varie spiegazioni senza avere dubbi sulla corretta successione dei comandi da impartire o sulle opzioni da scegliere in un determinato frangente.

Il testo è stato tradotto perfino negli esempi riportati, tanto è vero che, nelle videate pubblicate, compaiono i testi di esempio in italiano.

Chi ha a che fare spesso con Word, pertanto, può da oggi disporre di un volume davvero valido, in grado di introdurre l'utente ad argomenti, anche se complessi, con la massima semplicità possibile.

il compilatore sia meno "potente" del vetusto Basic del C/64?
(Nunzio Santini - Comiso)

No! Ci mancherebbe altro. C'è da dire, anzitutto, che è il Basic del C/64 ad esser "diverso" dal Basic standard, e non viceversa. Tutti i linguaggi Basic "ufficiali", infatti, annullano la stringa se si batte Return senza digitare nulla; o meglio, ad esser più precisi, considerano come risposta valida la sequenza di caratteri compresi tra il punto interrogativo (tipico dell'Input) e la cella video in cui si trova il cursore al momento della pressione del tasto Return. Se, poi, ne vogliamo fare una questione di comodità, bisogna tener conto che, con il C/64, siamo costretti a digitare la coppia di virgolette ("") nel caso in cui, effettivamente, vogliamo render nulla la stringa A$.

Se, comunque, vogliamo che alla stringa A$ rimanga associata la parola "Pippo" anche nel caso in cui si batta Return "a vuoto", il problema si risolve in modo semplicissimo, ricorrendo ad una variabile di comodo **B$**. Esempio:

```
A$ = "Pippo": B$ = A$
Input A$: if A$ = ""
then A$ = B$
```

Per quanto riguarda le altre tecniche, legate a trucchetti vari, ben noti nel C/64 (forzatura del buffer di tastiera e simili) ci stiamo attrezzando. Un po' di pazienza!

I sosia

Vi sono computer realmente portatili, che simulano il C/64, l'Ms - Dos, l'Amiga?
(Emanuele Giacometti - S.M. Sala)

Tempo fa esisteva il **C/64 executive** che, appunto, incorporava un C/64 (naturalmente!) un drive 1541 ed un mini monitor a colori. In effetti, più che un portatile, era un trasportabile, dal momento che pesava ben **11 chili** e non funzionava a batterie, ma richiedeva l'alimentazione a rete.

Di **Amiga portatili** non ho sentito parlare, ma dubito che (se non ci pensa mamma Commodore) ci sia qualcuno in grado di proporre dei "cloni": i circuiti Amiga sono brevettatissimi e l'imprudente "copiatore" non avrebbe vita facile.

Per ciò che riguarda il mondo **Ms - Dos** i portatili si sprecano: quasi ogni settimana ne viene proposto uno nuovo, sempre più potente del precedente. Funzionano a batterie (ricaricabili) la cui autonomia raggiunge le 4 ore; lo schermo è a cristalli liquidi e riesce a simulare una **VGA**. Il prezzo? molto vicino ai cinque milioni. E parliamo dei modelli più economici...

Giocare al Lotto

Ho scritto un programma per C/64, in grado di facilitare le giocate del Lotto, che è stato tempo fa pubblicato da una casa editrice vostra concorrente, e venduto su dischetto. Potrebbe interessarvi una versione migliorata dello stesso prodotto?
(S. N. Cerea)

No, e questo per due motivi. Anzitutto un nostro valido collaboratore ha realizzato un programma del tipo proposto, scritto in **C**, di notevole potenza; attualmente è in fase di prova e, quanto prima, verrà proposto ai nostri lettori.

Inoltre non vedo per quale motivo dovremmo divulgare un prodotto che ha già saturato la propria fascia di mercato

Penthouse per Amiga ed Ms - Dos

Chi ha avuto modo di ammirare le maliziose ragazze, sommariamente vestite, pubblicate sulla nota rivista americana **Penthouse**, da qualche tempo ha la possibilità di ritrovarsele anche sullo schermo del proprio **Amiga** oppure **Ms - Dos** compatibile.

12 immagini "particolari" (ma nulla di eccessivo, comunque) fanno parte di un programma che consente di spezzarle in tantissimi puzzle, da ricomporre a colpi di mouse. Chi ha poca pazienza, comunque, può ammirare le varie immagini (una per ciascun mese dell'anno) nella loro interezza.

Come ogni puzzle che si rispetti, anche **"Penthouse Electric Jigsaw"** (nome del programma in questione) consente di memorizzare la fase di ricostruzione allo scopo di riprenderla in un secondo momento.

La dimensione dei puzzle è programmabile (16 formati differenti) per rendere il "gioco" sempre più difficile.

SYSTEMS EDITORIALE PER TE

La voce
Aggiunge al C/64 nuovi comandi Basic che consentono sia di far parlare il computer, sia di farlo Cantare! Diversi esempi allegati.

Cassetta: L. 12000 - Disco: L. 15000

Raffaello
Un programma completo per disegnare, a colori, con il C/64: linee, cerchi, quadrati, eccetera. Valido sia per disegno a mano libera che geometrico.

Cassetta: L. 10000

Oroscopo
Devi solo digitare la data di nascita e le coordinate geografiche del luogo che ti ha dato i natali. Vengono quindi elaborate le varie informazioni (case, influenze dei segni astrali, eccetera) e visualizzato un profilo del tuo carattere. Valido per qualsiasi anno, è indicato sia agli esperti sia ai meno introdotti. E' allegata una tabella delle coordinate delle più note città italiane e l'elenco delle ore legali in Italia dal 1916 al 1978.

Cassetta: L. 12000 - Disco: L. 12000

Computer Music
Cassetta contenente numerosi brani di successo da far eseguire, in interrupt, al tuo C/64 sfruttando, fino in fondo, il suo generatore sonoro (SID).

Cassetta: L. 12000

Gestione Familiare
Il più noto ed economico programma per controllare le spese e i guadagni di una famiglia.

Cassetta: L. 10000 - Disco: L. 10000

Banca Dati
Il più noto ed economico programma per gestire dati di qualsiasi natura.

Cassetta: L. 10000 - Disco: L. 10000

Matematica finanziaria
Un programma completo per la soluzione dei più frequenti problemi del settore.

Cassetta: L. 10000 - Disco: L. 20000

Analisi di bilancio
Uno strumento efficace per determinare con precisione i calcoli necessari ad un corretto bilancio.

Cassetta: L. 10000 - Disco: L. 20000

Corso di Basic
Confezione contenente quattro cassette per imparare velocemente le caratteristiche delle istruzioni Basic del C/64 e i rudimenti di programmazione. Interattivo.

Cassetta: L. 19000

Corso di Assembler
Un corso completo su cassetta per chi ha deciso di abbandonare il Basic del C/64 per addentrarsi nello studio delle potenzialità del microprocessore 6502. Interattivo.

Cassetta: L. 10000

Logo Systems
Il linguaggio più facile ed intuitivo esistente nel campo dell'informatica; ideale per far avvicinare i bambini al calcolatore.
Diversi esempi allegati.

Cassetta: L. 6500

Compilatore Grafico Matematico
Uno straordinario programma compilatore, di uso semplicissimo, che permette di tracciare, sul C/64, grafici matematici Hi-Res ad altissima velocità. Esempi d'uso allegati.

Cassetta: L. 8000

Emulatore Ms-Dos e Gw-Basic
Un prodotto, unico nel suo genere, che permette di usare, sul C/64 dotato di drive, la sintassi tipica del più diffuso sistema operativo del mondo. Ideale per studenti.

Solo su disco: L. 20000

Emulatore Turbo Pascal 64
Permette di usare le più importanti forme sintattiche del linguaggio Turbo Pascal (anche grafiche!) usando un semplice C/64 dotato di drive.
Ideale per studenti.

Disco: L. 19000

Speciale drive
Questo speciale fascicolo costituisce una guida di riferimento per le unità a disco del C64/128.
Comprende anche un velocissimo turbo-disk più la mappa completa della memoria del drive.

Fascicolo + disco: L. 12000

Utility 1
Un dischetto pieno zeppo di programmi speciali per chi opera frequentemente con il drive.

Disco: L. 12000

Utility 2
Seconda raccolta di utility indispensabili per realizzare sofisticate procedure di programmazione.

Disco: L. 15000

Graphic Expander 128
Per usare il C/128 (in modo 128 e su 80 colonne) in modo grafico Hi-res. Aggiunge nuove, potenti istruzioni Basic per disegnare in Hi-Res con la massima velocità in modalità 80 colonne.

Disco: L. 27000

Directory
Come è noto, a partire dal N. 10 di "Software Club" (la rivista su disco per l'utente dei "piccoli" computer Commodore), vengono riportati tutti i listati, in formato C/64-C/128, pubblicati su "Commodore Computer Club".
In precedenza tali listati venivano inseriti, mensilmente, in un dischetto, di nome "Directory", che oltre ai programmi di C.C.C. ospitava decine di altri file tra cui musiche nell'interrupt, giochi, listati inviati dai lettori e altro.
Ogni disco, dal prezzo irrisorio, contiene quindi una vera miniera di software. Ordinando i dischetti di "Directory" si tenga conto che al N. 1 corrispondeva il contenuto del N. 34 di "Commodore Computer Club", al N. 2 il N. 35 e così via.

Ogni dischetto: L. 10000

Super Tot '64
La nuova e completa edizione del programma Tot 13 con tutti i sistemi di riduzione e di condizionamento.
Ampia sezione dedicata alla teoria.

fascicolo + disco: L. 15000

Amiga Totospeed
Finalmente anche per Amiga un programma orientato alla compilazione delle schedine totocalcio.
Fai tredici conil tuo Amiga.

disco: L. 20000

SYSTEMS EDITORIALE PER TE

Disk'o'teca
Grazie a questa nutrita raccolta di brani musicali potrete divertirvi ascoltando i migliori brani prodotti dai vostri beniamini, oltre a una serie di composizioni prodotte "in casa".
In omaggio un bellissimo poster di Sting.
Disco: L.15.000

Assaggio di primavera
Esclusivo!
In un'unica confezione potrete trovare ben due cassette di videogiochi assieme a un comodo e funzionale joystick.
Cassette: L. 15.000

LIBRI TASCABILI

64 programmi per il C/64
Raccolta di programmi (giochi e utilità) semplici da digitare e da usare. Ideale per i principianti. **(126 pag.)**
L. 4800

I miei amici C/16 e Plus/4
Il volumetto, di facile apprendimento, rappresenta un vero e proprio mini-corso di Basic per i due computer Commodore. Numerosi programmi, di immediata digitazione, completano la parte teorica. **(127 pag.)**
L. 7000

62 programmi per C/16, Plus/4
Raccolta di numerosi programmi, molto brevi e semplici da digitare, per conoscere più a fondo il proprio elaboratore.
Ideale per i principianti. **(127 pag.)**
L. 6500

Micro Pascal 64
Descrizione accurata della sintassi usata dal linguaggio Pascal "classico". Completa il volume un programma di emulazione del PL/O sia in formato Microsoft sia in versione C/64 (da chiedere, a parte, su disco). **(125 pag.)**
L. 7000

Dal registratore al Drive
Esame accurato delle istruzioni relative alle due più popolari periferiche del C/64. Diversi programmi applicativi ed esempi d'uso. (94 pag.)
L. 7000

Il linguaggio Pascal
Esame approfondito della sintassi usata nel famoso compilatore. (112 pag.)
L. 5000

Simulazioni e test per la didattica
Raccolta di numerosi programmi che approfondiscono e tendono a completare la trattazione già affrontata sul precedente volume. **(127 pag.)**
L. 7000

Dizionario dell'Informatica
Dizionario inglese-italiano di tutti i termini usati nell'informatica. (Edizione completa). **(385 pag.)**
L. 10000

Word processing: istruzioni per l'uso
Raccolta delle principali istruzioni dei più diffusi programmi di w/p per i sistemi Ms-Dos: Word-Star, Samna, Multimate Advantage, Word 3. **(79 pag.)**
L. 5000

Unix
Un volumetto per saperne di più sul sistema operativo professionale per eccellenza.
Un necessario compendio per l'utente sia avanzato che inesperto **(91 pag.)**
L. 5000

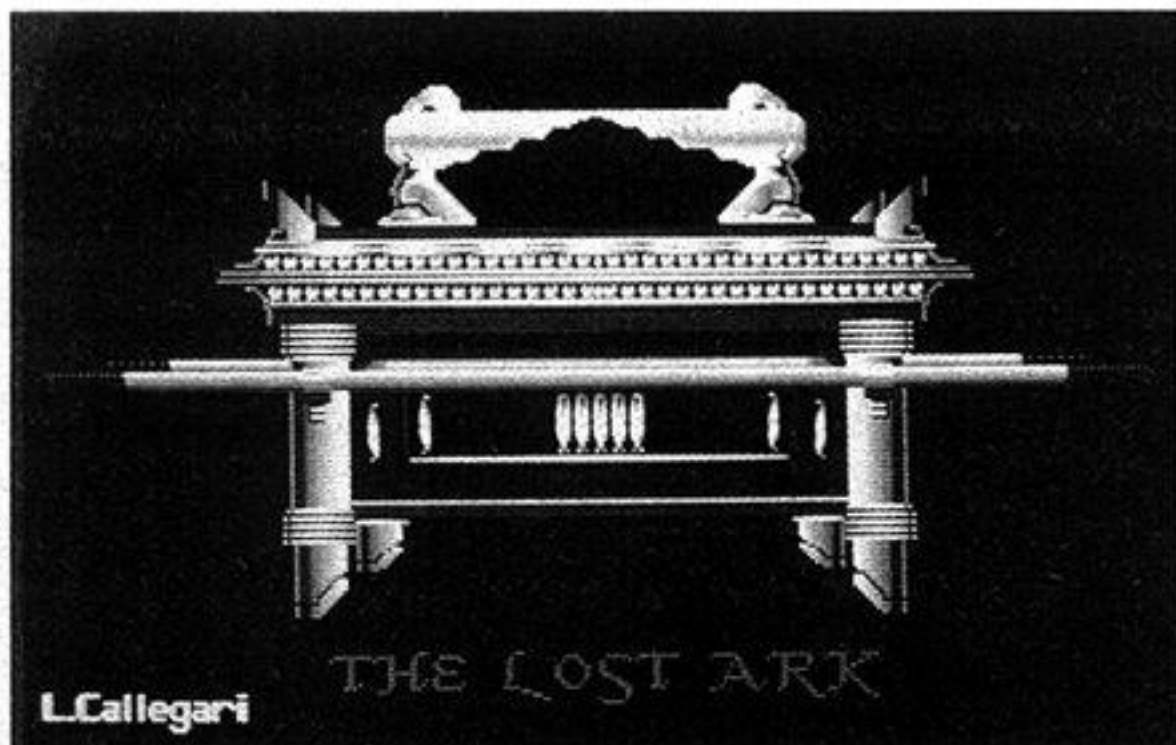
ABBONAMENTO
Commodore Computer Club
11 fascicoli: L. 60.000

ARRETRATI
Ciascun numero arretrato di C.C.C. L. 6.000

Come richiedere i prodotti Systems
Coloro che desiderano procurarsi i prodotti della Systems Editoriale devono inviare, oltre alla cifra risultante dalla somma dei singoli prodotti, L. 3500 per spese di imballo e spedizione, oppure L. 6000 se si desidera la spedizione per mezzo raccomandata.
Le spese di imballo e spedizione sono a carico della Systems se ciascun ordine è pari ad almeno L. 50000.
Per gli ordini, compilare un normale modulo di C/C postale indirizzato a:

C/C Postale N. 37 95 22 07
Systems Editoriale Srl
Via Mosè, 22
20090 Opera (MI)

Non dimenticate di indicare chiaramente, sul retro del modulo (nello spazio indicato con "Causale del versamento"), non solo il vostro nominativo completo di recapito telefonico, ma anche i prodotti desiderati ed il tipo di spedizione da effettuare.
Per sveltire la procedura di spedizione sarebbe opportuno inviare, a parte, una lettera riassuntiva dell'ordine effettuato, allegando una fotocopia della ricevuta del versamento.
Chi volesse ricevere più celermente la confezione deve inviare la somma richiesta mediante assegno circolare, oppure normale assegno bancario (non trasferibile o barrato due volte) intestato a:

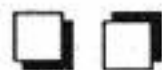
Systems Editoriale
Milano

e che ha, quindi, scarsa probabilità di un certo successo commerciale. Le consiglio, pertanto, di continuare la collaborazione con la stessa casa editrice che ha già provveduto a commercializzare il programma in oggetto.

❑❒

Fusioni video

Vorrei che, durante la consueta ricezione di programmi televisivi, alcuni messaggi generati con il mio computer appaiano in sovraimprtessione (tipo pagina 777 televideo) o sovrapposti alle immagini. Come posso operare per raggiungere lo scopo?

(anonimo 64ista)

E' indispensabile che l'uscita video del computer sia collegabile ad un **miscelatore video**. Quest'ultimo è un apparecchio che, da un po' di tempo, incomincia ad avere un certo successo tra gli appassionati di **videoregistrazione**. Purtroppo è un accessorio non proprio economico, dal momento che si rivolge ad un'utenza quasi professionale. Il problema della miscelazione dei segnali video, infatti, richiede l'utilizzo di componenti elettronici piuttosto particolari, certamente più costosi di quelli necessari per un banale miscelatore audio.

Margherite nazionali

La stampante in mio possesso è dotata di margherita americana. Posso usarla anche con un computer dotato di tastiera italiana?

(anonimo de Roma)

Le stampanti a margherita consentono di ottenere una qualità di stampa eccellente, ma presentano alcuni problemi. Anzitutto, non è possibile riprodurre schermate grafiche dal momento che le immagini non di testo vengono simulate, dalle stampanti ad aghi, mediante la corrispondenza tra **dot** (ago della testina di stampa) e **pixel** (puntino elementare dello schermo video). Se il puntino c'è, in corrispondenza l'ago della stampante viene "spinto" per farlo apparire anche su carta. La *spinta* non avviene se il pixel risulta, invece, "spento". L'altro svantaggio delle stampanti a margherita è rappresentato dal fatto che, volendo cambiar tipo di carattere, è necessario sostituire la margherita stessa. Ogni suo "petalo", infatti, contiene un carattere e siccome il numero di petali è ben definito (a causa delle dimensioni della margherita) ne consegue che i caratteri riproducibili da ogni margherita non sono numerosi.

In base ad una convenzione internazionale, comunque, ogni carattere trova posto in corrispondenza di un ben determinato petalo, che la periferica rintraccia (automaticamente) non appena viene messa in funzione. Il tipo di tastiera del computer, quindi, non c'entra per nulla. Semmai devi preoccuparti di utilizzare margherite i cui petali posseggano, effettivamente, le vocali accentate italiane.

Quanti Amiga?

Amiga ha ancora successo nel mondo, oppure la Commodore pensa di interromperne la produzione?

(da varie lettere)

No, assolutamente. Anche se non vogliamo credere all'ottimismo di casa Commodore, basterà fare un paio di ragionamenti per convincersene egualmente: già prima dello scorso Natale erano stati venduti, nel mondo, oltre **due milioni di computer della serie Amiga**. Con l'introduzione del modello 3000 la Commodore cerca, oggi, di imporsi anche nella fascia professionale; il recente abbassamento dei costi di produzione (e dei prezzi al pubblico) consente uno sviluppo della domanda sempre maggiore; i programmi e gli accessori disponibili (anche non originali) proliferano senza fine. Solo un fabbricante ingenuo potrebbe pensare di abbandonare una linea di prodotti che si vende "da sola", e con notevole profitto, senza alcuna difficoltà.

di Rodolfo Facchinetti

IL RASTER SCONOSCIUTO

Cinque routines rilocabili, utili per la manipolazione "pesante" di uno sprite con il C/64

L'articolo è dedicato agli ex principianti (ed aspiranti esperti) e non vuole essere il solito resoconto in cui si spiega come funziona un certo programma, ma illustra il metodo "scientifico" seguito dall'autore per raggiungere lo scopo, partendo da alcune premesse teoriche.

Tutto è iniziato quando, scrivendo un videogioco in Basic, fu presa la decisione di riscriverlo in l.m. allo scopo di meglio imparare questo ardito linguaggio.

Nel game, arrivati ad un certo punto, uno sprite veniva "colpito", esplodendo.

L'effetto sarebbe risultato più interessante dividendo lo sprite in sei parti, allontanantisi in altrettante direzioni diverse.

Una facile soluzione al problema sarebbe stata quella di utilizzare sei sprites differenti. Gli altri sprites, però, risultavano già impegnati nel gioco e dovevano restare visualizzati permanentemente sullo schermo.

L'uso di **un solo sprite** (e, pertanto, l'uso del Raster Register) si rendeva quindi indispensabile.

Chi non vuole sorbirsi la parte teorica, può saltare a piè pari l'intero articolo, soffermandosi solo sui consigli su come caricare, ed attivare, le varie routines l.m. rilocabili pubblicate in queste pagine.

Dunque, per non partire proprio da zero, una scorsa ai fascicoli arretrati di C.C.C. è stata preziosa: sia il programma di Gianluca Venturini (c.c.c. n. 63) **Come gestire 64 sprites** (il quale è una miglioria apportata al programma di Michele Maggi, **La moltiplicazione degli sprites**, entrambi pubblicati sugli inserti Campus) hanno consentito di iniziare il lavoro.

Vi suggeriamo, ora, di seguire quanto diremo, ma prima dovrete caricate in memoria un Assemblatore perchè dovremo cambiare il programma in l.m. prima accennato. Caricato il programma pubblicato sul N. 63, ed impartito il Run, vedremo i 64 sprites, ben allineati, che si stanno colorando di nero. Bloccate il programma con Run Stop / Restore quando si è colorato a metà il primo sprite della terza fila.

Pulite il video ancora con Run Stop / Restore e poi impartite **List -50.** Cancellate il programma Basic con **New** e premete il tasto Return dopo esservi posizionati sulle linee 50 e 60, che gestiscono la posizione orizzontale degli sprites, la loro accensione e la Sys di attivazione. Con un nuovo Run, i 64 sprite riappariranno di nuovo, confermando che... nulla è cambiato. Ora, in linea 60, spegnete tutti gli sprites meno uno, mediante **Poke 53269, 1** (relativa all'abilitazione degli sprites). Un altro Run visualizzerà una sola colonna di sprites, ma sappiamo benissimo che, in effetti, si tratta di **un solo sprite** (ed esattamente il n. 0) **visualizzato 8 volte**, una sotto l'altra, con 8 figure diverse, e lo prova il fatto che solo alcune sono colorate di nero; ciò dipende da quando avete bloccato il programma all'inizio.

Bene, visto che trattiamo un solo sprite, è inutile dare la coordinata orizzontale a tutti, quindi modificate la linea 50 con **Poke 53248, 157**: la colonna di sprites si sposterà verso il centro dello schermo, a ribadire che si tratta di un solo sprite, pur se visualizzato 8 volte. Premete Run Stop / Restore ed attivate l'assemblatore, perchè ora modificheremo il programma l.m.

Disassemblando da **C0D6**, avremo sullo schermo le ultime tre routines inserite nell'Interrupt, cioè **SPR6 - SPR7** e metà di **SPR8**. Eliminiamo **SPR7** e **SPR8** riempiendo la memoria di zeri, a partire da **C0EC** (comando: **F C0EC C120 00**), e torniamo a **C0D6**. E' rimasta la sola

```
10 rem prova 6 immagini
20 print chr$(147)
30 r=49152:v=53248
40 read a:ifa=-1then 55
50 poker,a:r=r+1:b=b+a:goto 40
55 r=r-49152:read a,aa
60 ifb<>a then print"errore":end
70 ifr<>aa then print"errore nel
          numero dei data":end
80 poke2040,250:pokev+39,1
90 pokev,140:pokev+1,140
100 pokev+21,1:sys49152:end
1000 data 162,006,160,000,173,000,208
1001 data 153,176,002,200,173,001,208
1002 data 153,176,002,200,202,208,239
1003 data 234,169,127,141,013,220,169
1004 data 107,141,020,003,169,192,141
1005 data 021,003,173,017,208,041,127
1006 data 141,017,208,169,129,141,026
1007 data 208,169,000,141,018,208,234
1008 data 169,000,133,252,206,176,002
1009 data 206,177,002,206,179,002,206
1010 data 180,002,238,181,002,238,182
1011 data 002,206,183,002,238,185,002
1012 data 238,186,002,238,187,002,169
1013 data 048,133,249,032,179,230,198
1014 data 249,208,249,198,252,208,211
1015 data 096,234,173,025,208,041,001
1016 data 208,003,076,188,254,141,025
1017 data 208,234,165,250,208,014,169
1018 data 249,141,248,007,162,000,160
1019 data 001,230,250,076,200,192,201
1020 data 001,208,003,162,002,160,003
1021 data 230,250,076,200,192,201,002
1022 data 208,009,162,004,160,005,230
1023 data 250,076,200,192,201,003,208
1024 data 009,162,006,160,007,230,250
1025 data 076,200,192,201,004,208,009
1026 data 162,008,160,009,230,250,076
1027 data 200,192,169,000,133,250,162
1028 data 010,160,011,234,184,176,002
1029 data 141,000,208,185,176,002,141
1030 data 001,208,238,248,007,076,049
1031 data 234,-1,29611,218

ready.
```

```
Programma 1

C000 SEI
C001 LDA#$27
C003 STA$0314
C006 LDA#$C0
C008 STA$0315
C00B CLI
C00C RTS
. C010
STA$D011
C027 LDA$FA
C029 BNE$C037
C02B LDX#$9D
C02D LDY#$60
C02F JSR$C044
C032 INC$FA
C034 JMP$EA31
C037 LDX#$9D
C039 LDY#$B0
C03B JSR$C044
C03E DEC$FA
C040 JMP$EA31
C043 NOP
C044 STX$D000
C047 STY$D001
C04A RTS
```

```
Programma 2

C000 SEI
C001 LDA#$1A
C003 STA$0314
C006 LDA#$C0
C008 STA$0315
C00B LDA$D011
C00E AND#$7F
C013 LDA#$81
C015 STA$D01A
C018 CLI
C019 RTS
C01A LDA$D019
C01D AND#$01
C01F BNE$C024
C021 JMP$FEBC
C024 STA$D019
C03B JSR$C044
C027 Prosegue
con il 1.
```

I due listati di riferimento

SPR6, da modificare: cioè l'uscita (JMP FEBC) in **JMP EA31**, per consentire all'Interrupt di continuare per la sua strada; trascivere zero nel Raster per un nuovo ciclo di Interrupt ed infine eliminare le tre prime istruzioni con un trasferimento (comando: **T C0DD C100 C0D6**).

SPR6, ora, è così formata:

LDX# $B0
LDY# $E8
JSR $C003
LDA# $00
STA $D012
JMP $EA31

Usciamo dall'Assemblatore e diamo Run. Le figure visualizzate sono solo sei, proprio quante ne occorrono, dal momento che il nostro obiettivo è rappresentato da sei figure, che formano un unico sprite, che si muovono in sei diverse direzioni. Poichè la routine nell'Interrupt è sempre attiva, prima di modificare il programma l.m. è meglio disattivarla con Run Stop / Restore. Torniamo quindi a C000. La subroutine **Visual** mette il registro **X** nella coordinata verticale di tutti gli sprites (ma ora non serve più) e cambia il puntatore a tutti gli sprites (ed anche questo non serve più). Quindi modifichiamo la subroutine da C003 che deve risultare:

STX $D001
STY $07F8
RTS

"Uscite" e date Run. Nulla è cambiato, ma le istruzioni poste tra i due **RTS** sono ora inutili, quindi eliminiamole. Disassemblate da C000 e riempite, con valori nulli, le locazioni citate (comando: **F C00A C03B 00**).

Nonostante abbiamo accorciato, e di molto, la routine posta nell'Interrupt e Visual, il programma funziona ancora benissimo; questo confortante particolare ci informa che la routine di Raster non è stata influenzata dalla lunghezza della nostra routine. Ne consegue che i **cicli macchina**, che è possibile eseguire tra due scatti del Raster, sono compresi tra zero ed un valore massimo.

Le risposte alle domande (sui cicli macchina e sulla lunghezza massima di una routine da inserire nel ciclo di Interrupt) le troviamo nell'articolo di Lorenzo Emilitri "Tutti i tempi del video", pubblicato tempo fa su Campus.

Leggendo l'articolo, si può determinare il numero di cicli macchina da eseguire in ogni linea di Raster; sarà quindi sufficiente(!) calcolare il numero di cicli richiesti dalla totalità delle istruzioni elaborate nel corso della routine posta in Interrupt per stabilire se il valore massimo viene superato oppure no.

Tornando al nostro programma, dovremmo inserire le sei immagini nello stesso punto, per poi allontanarle gradualmente. Cominciamo quindi ad avvicinare un'immagine alla successiva. Poichè, mentre lavoreremo, la routine in Interrupt dovrà essere attiva, è meglio salvare ciò che abbiamo finora digitato, perchè è molto facile andare in tilt nel corso delle manipolazioni dei puntatori.

Date Run e disassemblate a partire da **C06D**, lasciando visualizzati gli sprite sullo schermo. Cercate di leggere (gli sprite, infatti, potrebbero "coprire" il testo) le tre prime routine che girano nell'Interrupt; modificando la posizione verticale della terza immagine, l'istruzione **C09B** diverrà **A2 64**. Il terzo "sprite" si è alzato di una riga. Ancòra: A2 **63**. Si è alzato ancòra. Di nuovo A2 **62** e poi A2 **61** e poi A2 60. Alt! Che cos'è successo? L'immagine è sparita!

La risposta allo strano fenomeno è semplice e logica. Nella routine precedente, **SPR2**, abbiamo immesso nel Raster il valore $60 (in C08C) e quindi il Raster è scattato a $60 (logico, no?) E' stata eseguita la terza routine (SPR3), ma al momento di "saltare" alla subroutine, il Raster, proseguendo la sua corsa, ha già oltrepassato la riga $60, si trova alla riga $61 e non può quindi gestire un'immagine posta sulla precedente riga $60. Poichè non ha saltato solo la prima riga dell'immagine, ma l'**intera** immagine, è evidente che (da qualche parte nella memoria) esiste un controllo per cui se il Raster ha oltrepassato la coordinata verticale posta a D001 (sprite 0) l'immagine non sarà visualizzata.

Bene, visto che a $61 l'immagine c'era, proviamo a ripristinare la differenza tra lo scatto del Raster e la coordinata verticale dell'immagine, cambiando però il valore per lo scatto del Raster con **C08C A9 5F**. Nonostante ciò, l'immagine non appare ma, diminuendo ancora i valori, la secon-

```
1000 print"        preroutine"
1010 print"va lanciata prima delle"
1020 print"routines effetto"
1030 print"con la sintassi sys xxx,n"
1040 print"xxx = locazione d'inizio"
1050 print"n = numero dello sprite"
1060 print"e' rilocabile"
1070 input"locazione di inizio";x
1080 fori=xtox+128:read a:y=y+a
1090 pokei,a:next
1095 ify=15623thenprint"fine="x+128:end
1096 print"errore":end
1100 data 032,241,183,142,167,002,189
1101 data 039,208,141,168,002,189,248
1102 data 007,141,169,002,138,010,170
1103 data 142,177,002,189,000,208,141
1104 data 170,002,189,001,208,141,171
1105 data 002,169,001,174,167,002,240
1106 data 004,010,202,208,252,141,174
1107 data 002,073,255,141,175,002,173
1108 data 023,208,045,174,002,141,172
1109 data 002,173,029,208,045,174,002
1110 data 141,176,002,173,028,208,045
1111 data 174,002,141,173,002,173,016
1112 data 208,045,174,002,141,178,002
1113 data 173,027,208,045,174,002,141
1114 data 179,002,169,000,133,253,173
1115 data 169,002,024,106,102,253,106
1116 data 102,253,133,254,160,000,177
1117 data 253,153,192,002,200,192,063
1118 data 208,246,096,-1,15623

ready.
```

da immagine diverrà la copia esatta della terza.

E' quindi probabile che la precedente teoria è errata, almeno in parte. Comunque è chiaro che in questo modo non risolveremo il problema. Proviamo, dunque, un'altra strada.

Cancellate l'intero programma L.M. (comando: **F C000 COFF 00**) e assemblate il programma 1 (del riquadro in questa pagina) nelle locazioni indicate.

Inserite nel programma Basic l'istruzione **Poke 250, 0** per azzerare il flag (interruttore) **$FA**; poi date Run. Dovrebbero apparire due immagini dello stesso sprite, pur se lampeggianti. Il perchè è presto detto. Nel modo normale, il pennello elettronico rinfresca l'immagine ogni cinquantesimo di secondo. In questo modo, invece, la frequenza si riduce alla metà, in quanto viene visualizzata una sola immagine per ogni passaggio del Raster, ed il nostro occhio nota la differenza. Comunque potrebbe essere un effetto gradevole ed insperato.

Così facendo possiamo avvicinare le due immagini. Disassembliamo da C027 lasciando in funzione il programma (a proposito, l'avete salvato?) Modifichiamo il valore di **Y** in una delle due routine, avvicinando un'immagine all'altra. Sovrapponete parzialmente le due immagini (**C039 A0 6A**) e notate che le parti sovrapposte non lampeggiano più. Sovrapponete completamente le due immagini e... perfetto, un primo passo è fatto. (Bisognerà poi vedere quando saranno 6 ...eh eh eh)

Spostando orizzontalmente e verticalmente un'immagine, notiamo che ci sono delle interferenze (chiamiamole così). Per eliminarle, modifichiamo il programma (che è così elementare che non necessita di spiegazioni) in modo che la routine non venga eseguita dal normale Interrupt, ma dall'Interrupt di Raster; la modifica è riportata nel secondo riquadro di questa stessa pagina (ma prima eseguite un provvidenziale Run / Stop + Restore).

Con Run, ora, le "interferenze" sono sparite. Abbiamo scoperto una grande verità: la frequenza del Raster non è uguale a quella dell'Interrupt. E' il momento di provare la stessa tecnica con sei immagini.

Le sei immagini

Per questo scopo ci sarà utile il programma dal nome **Prova 6 immagini**, pubblicato soltanto sotto forma di **Data** a causa della sua lunghezza. Inutili sarebbero le spiegazioni del procedimento seguito, perchè non è che uno sviluppo del precedente listato.

Dopo averlo caricato, però, rimarrete forse delusi perchè, dividendo per sei la frequenza del Raster, le immagini subiscono un lampeggio accentuato, inaccettabile. Morale: neppure in questo modo il problema è risolto, pur se le esperienze effettuate costituiscono un'ottima base per futuri sviluppi.

Nonostante, però, il parziale insuccesso delle tecniche effettuate, possiamo affermare che abbiamo creato, magari involontariamente, alcuni effetti non meno interessanti di quelli cercati:

1 Modifica lenta del colore dello sprite (se, inoltre, il colore è quello dello sfondo, lo sprite scompare.)

2 Taglio dello sprite in due parti: la prima esce a destra, l'altra a sinistra.

Dal momento che le routine (e altre dello stesso tipo) abbisognano delle stesse informazioni riguardanti lo sprite da "ammazzare" si è preferito raggrupparle tutte in un unico listato dal nome **Preroutine**, che dovrà essere lanciato, per primo, con la sintassi:

preroutine SYS xxx, numero sprite
cambia colore SYS xxx, colore cambio
taglia sprite SYS xxx

Le routine, "ovviamente", sono tutte rilocabili.

A cambiare il colore dello sprite pensano due routinette poste alternativamente nell'Interrupt e selezionate dal flag $FA (l'abbiamo già visto). Una è eseguita quando il Raster è a 00 (contiene il colore del cambio) mentre l'altra (contiene il colore originale) quando il Raster è all'altezza dello sprite. Se il Raster scatta quando ha già cominciato a visualizzare lo sprite, ne cambierà il colore col risultato citato.

Per tagliare lo sprite, basta far scattare il Raster a metà sprite e poi immettere valori diversi nella coordinata orizzontale tramite le due solite routinette. Queste sembrano complicate soltanto perchè devono considerare anche gli sprites espansi e multicolori e la posizione sullo schermo oltre il 255mo pixel.

Dalla routine **Taglia sprite** è uscita una nuova teoria: ad ogni riga di Raster, o almeno ad ogni Interrupt, viene controllata la coordinata orizzontale degli sprites. Bella teoria, vero?

Speriamo che almeno questa duri...

LA MORTE DEGLI SPRITES

Probabilmente, dopo averlo "maltrattato", lo sprite sarà irriconoscibile ed i suoi attributi (colore, coordinate, espansione ecc.) risulteranno alterati rispetto all'originale. Per ridare allo sprite le sue originarie peculiarità, si rende necessario salvare i suoi attributi in una zona di memoria al riparo dalle "intemperie"; quella che fa al caso nostro la troviamo in pagina 2, da **$02A7** fino a **$02FF**.

La routine, chiamata Preroutine, dovrà essere lanciata per prima con la sintassi:

SYS xxx, x

...in cui **xxx** è la locazione d'inizio e **x** il numero dello sprite, da scegliere tra 0 e 7.

Da essa si dovranno attingere i dati da usare nelle routines di effetto. Alla fine di queste routines, la preroutine dovrà essere lasciata integra per poterla riutilizzare in altre routines - effetto.

Le routines pubblicate sono:

1. Cambio del colore
2. Taglio in due pezzi
3. Disintegrazione
4. Schiacciamento
5. Dipartizione dell'anima

Nella **prima**, il colore dello sprite viene lentamente cambiato utilizzando il Raster Register. Inizialmente, quando il pennello elettronico si trova a $00, viene eseguita una breve elaborazione con la

```
100 poke 2040,0 :rem banco sprite 0
110 poke 53248,130:rem pos. x sprite 0
120 poke 53249,130:rem pos. y sprite 0
130 poke 53287,0:rem colore sprite 0
140 poke 53269,255:rem abilitaz.sprites

ready.
```

Il micro - programma Sprite

```
1000 print"      cambia colore"
1010 print"sintassi: sys xxx,c"
1020 print"xxx = locazione d'inizio"
1030 print"c = colore del cambio"
1040 print"e' rilocabile"
1050 input"locazione di inizio";x
1051 fori=xtox+175:read a:y=y+a
1052 pokei,a:next
1053 ify=23188thenprint"fine="x+175:end
1054 print"errore":end
1100 data 032,241,183,134,252,169,000
1101 data 133,250,120,165,021,141,021
1102 data 003,024,165,020,105,099,141
1103 data 020,003,144,003,238,021,003
1104 data 169,129,141,026,208,173,017
1105 data 208,041,127,141,017,208,088
1106 data 173,171,002,133,251,160,022
1107 data 173,172,002,240,002,160,043
1108 data 234,230,251,169,129,141,004
1109 data 212,162,064,032,179,238,202
1110 data 208,250,238,001,212,136,208
1111 data 235,032,138,255,169,000,141
1112 data 026,208,141,004,212,174,167
1113 data 002,165,252,157,039,208,096
1114 data 234,173,025,208,041,001,208
1115 data 003,076,188,254,141,025,208
1116 data 165,250,208,031,169,000,141
1117 data 018,208,173,173,002,240,006
1118 data 077,028,208,141,028,208,174
1119 data 167,002,173,168,002,157,039
1120 data 208,230,250,076,188,254,234
1121 data 165,251,141,018,208,173,173
1122 data 002,240,006,077,028,208,141
1123 data 028,208,174,167,002,165,252
1124 data 157,039,208,198,250,076,049
1125 data 234,-1,23188

ready.
```

quale si immette il colore del cambio nel registro colore dello sprite. Quando, poi, il Raster raggiunge la coordinata verticale dello sprite, nel registro colore viene immesso il colore originale, e lo sprite viene visualizzato in quest'ultimo colore. Spostando l'interrupt di Raster (aumentandone il valore), il colore originale sarà immesso nel registro quando la visualizzazione dello sprite è già cominciata con il colore del cambio. Aumentando gradatamente il valore per lo scatto del Raster si avrà l'effetto desiderato.

Nella **seconda** routine lo sprite viene tagliato a metà; poi la parte superiore esce dal video a sinistra, mentre la parte inferiore esce da destra. Anche in questa routine è usato il Raster Register facendo scattare l'interrupt due volte ogni passaggio del pennello elettronico: il primo interrupt all'inizio, cioè a $00; il secondo quando il Raster è giunto a metà dello sprite. Sono quindi eseguite, alternativamente, due piccole routines (una ogni interrupt) con le quali la coordinata orizzontale dello sprite è spostata verso destra, o verso sinistra, ottenendo ciò che si voleva.

```
1000 print"  taglio in due pezzi"
1010 print"sintassi: sys xxx"
1020 print"xxx = locazione d'inizio"
1030 print"e' rilocabile"
1040 input"locazione di inizio";x
1041 fori=xtox+279:read a:y=y+a
1042 pokei,a:next
1043 ify=35147thenprint"fine="x+279:end
1044 print"errore":end
1100 data 169,017,141,004,212,169,000
1101 data 133,249,133,250,133,251,173
1102 data 170,002,141,052,003,141,053
1103 data 003,173,171,002,105,009,174
1104 data 172,002,240,002,105,010,141
1105 data 054,003,173,178,002,240,002
1106 data 169,001,133,247,133,248,234
1107 data 120,165,021,141,021,003,024
1108 data 165,020,105,175,141,020,003
1109 data 144,003,238,021,003,169,001
1110 data 141,026,208,173,017,208,041
1111 data 127,141,017,208,088,234,165
1112 data 249,240,002,016,024,173,052
1113 data 003,141,001,212,208,012,165
1114 data 247,208,006,165,249,208,057
1115 data 230,249,198,247,206,052,003
1116 data 234,165,249,048,034,160,255
1117 data 165,248,240,002,160,096,204
1118 data 053,003,208,012,192,255,240
1119 data 006,165,249,208,024,198,249
1120 data 230,248,238,053,003,173,053
1121 data 003,141,001,212,160,002,032
1122 data 179,238,136,208,250,240,178
1123 data 234,032,138,255,169,000,141
1124 data 026,208,141,004,212,096,234
1125 data 173,025,208,041,001,208,003
1126 data 076,188,254,141,025,208,165
1127 data 250,208,045,173,054,003,141
1128 data 018,208,165,247,240,011,173
1129 data 016,208,013,174,002,141,016
1130 data 208,208,009,173,016,208,045
1131 data 175,002,141,016,208,174,177
1132 data 002,173,052,003,157,000,208
1133 data 230,250,076,188,254,234,169
1134 data 000,141,018,208,165,248,240
1135 data 011,173,016,208,013,174,002
1136 data 141,016,208,208,009,173,016
1137 data 208,045,175,002,141,016,208
1138 data 174,177,002,173,053,003,157
1139 data 000,208,198,250,076,049,234
1140 data -1,35147

ready.
```

Con la **terza** routine si disintegra lo sprite modificando i suoi bytes effettuando un **And** con un valore sempre decrescente, finchè questo giunge a zero, momento in cui lo sprite scompare.

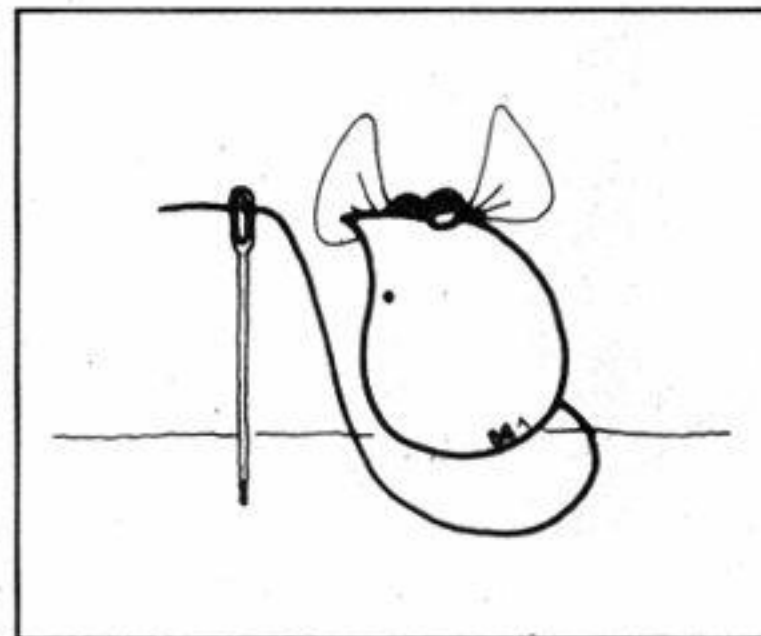
Nella **quarta** routine lo sprite viene compresso verso il centro spostando metodicamente una riga di bytes in quella sottostante, fino a raggiungere la riga centrale. Nella **quinta** ed ultima routine viene usato di nuovo il Raster Register, adoperando però un solo interrupt. E' usato il Raster, al posto del normale Interrupt, per evitare un'imperfetta visualizzazione.

Due routines, eseguite alternativamente, mostrano due immagini dello stesso sprite, una nel colore originale e

```
1000 print"   disintegrazione"
1010 print"sintassi: sys xxx"
1020 print"xxx = locazione d'inizio"
1030 print"e' rilocabile"
1040 input"locazione di inizio";x
1041 fori=xtox+ 54:read a:y=y+a
1042 pokei,a:next
1043 ify= 9274thenprint"fine="x+54 :end
1044 print"errore":end
1100 data 169,017,141,004,212,169,000
1101 data 133,250,133,252,234,160,063
1102 data 230,252,230,252,198,250,198
1103 data 250,136,185,192,002,037,250
1104 data 037,251,145,253,141,001,212
1105 data 230,251,166,252,202,208,253
1106 data 152,208,233,165,250,208,219
1107 data 169,016,141,004,212,096
1108 data -1,9274

ready.
```

l'altra bianca ("anima"). Spostando l'anima verso l'alto si avrà l'effetto desiderato.

Il programmino **Sprite** vi servirà se non avete sprite da manipolare; fa apparire, più o meno al centro del video, uno sprite (che punta ad un gruppo di locazioni in pagina 0) colorato in nero. Caricando, e lanciando, questa routine **prima** di caricare e lanciare le routines l.m. rilocabili, sarà possibile esaminare un primo, sommario effetto. Ovvio che i lettori più in gamba, cui è destinato l'articolo, dovranno completare le varie operazioni per proprio conto.

In ogni routine è anche inserito un piccolo effetto sonoro, volutamente semplicissimo per non appesantire le già lunghe routines.

```
1000 print"dipartizione dell'anima"
1010 print"sintassi: sys xxx"
1020 print"xxx = locazione d'inizio"
1030 print"e' rilocabile"
1040 input"locazione di inizio";x
1041 fori=xtox+137:read a:y=y+a
1042 pokei,a:next
1043 ify=18049thenprint"fine="x+137:end
1044 print"errore":end
1100 data 169,017,141,004,212,169,000
1101 data 133,250,141,018,208,173,171
1102 data 002,133,251,234,120,165,021
1103 data 141,021,003,024,165,020,105
1104 data 076,141,020,003,144,003,238
1105 data 021,003,169,129,141,026,208
1106 data 088,234,162,016,032,179,238
1107 data 202,208,250,165,251,073,255
1108 data 141,001,212,198,251,208,237
1109 data 032,138,255,169,000,141,026
1110 data 208,141,004,212,096,234,173
1111 data 025,208,041,001,208,003,076
1112 data 188,254,141,025,208,165,250
1113 data 208,022,172,177,002,165,251
1114 data 153,001,208,172,167,002,169
1115 data 001,153,039,208,230,250,076
1116 data 049,234,234,172,177,002,173
1117 data 171,002,153,001,208,172,167
1118 data 002,173,168,002,153,039,208
1119 data 198,250,076,049,234
1120 data -1,18049

ready.
```

```
1000 print"   schiacciamento"
1010 print"sintassi: sys xxx"
1020 print"xxx = locazione d'inizio"
1030 print"e' rilocabile"
1040 input"locazione di inizio";x
1041 fori=xtox+172:read a:y=y+a
1042 pokei,a:next
1043 ify=25655thenprint"fine="x+172:end
1044 print"errore":end
1100 data 174,167,002,169,011,157,248
1101 data 007,169,017,141,004,212,169
1102 data 010,133,249,169,002,133,250
1103 data 162,032,160,029,169,003,133
1104 data 251,189,192,002,141,001,212
1105 data 025,192,002,157,192,002,202
1106 data 136,198,251,208,238,162,032
1107 data 160,035,198,250,208,226,198
1108 data 169,208,252,234,169,027,133
1109 data 252,169,026,133,174,169,029
1110 data 133,175,169,036,133,247,169
1111 data 033,133,248,166,174,164,175
1112 data 189,192,002,153,192,002,169
1113 data 000,157,192,002,198,174,198
1114 data 175,166,247,164,248,189,192
1115 data 002,153,192,002,169,000,157
1116 data 192,002,230,247,230,248,198
1117 data 252,208,214,032,179,238,198
1118 data 169,208,249,198,249,208,140
1119 data 234,169,000,141,004,212,141
1120 data 224,002,141,222,002,032,179
1121 data 238,198,169,208,249,174,177
1122 data 002,169,255,157,001,208,160
1123 data 000,177,253,153,192,002,200
1124 data 192,063,208,246,096
1125 data -1,25655

ready.
```

```
    PREROUTINE    SYS xxx,n

RILOCABILE

C000 JSR $B7F1    Preleva il parametro.
C003 STX $02A7    Numero dello sprite.
C006 LDA $D027,X
C009 STA $02A8    Colore.
C00C LDA $07F8,X
C00F STA $02A9    Puntatore.
C012 TXA          Moltiplica per due il
C013 ASL A        numero dello sprite;
C014 TAX          serve a trovare le
C015 STX $02B1    locazioni delle coord.
C018 LDA $D000,X
C01B STA $02AA    Coordinata orizzontale
C01E LDA $D001,X
C021 STA $02AB    Coordinata verticale.
C024 LDA #$01
C026 LDX $02A7
C029 BEQ $C02F
C02B ASL A        Attiva in $02AE il bit
C02C DEX          relativo allo sprite
C02D BNE $C02B    e in $02AF il suo
C02F STA $02AE    complemento (inverso).
C032 EOR #$FF
C034 STA $02AF
C037 LDA $D017    Espansione verticale.
C03A AND $02AE    $02AC=00 non espanso
C03D STA $02AC    $02AC<>00 e' espanso
C040 LDA $D01D
C043 AND $02AE    Espansione orizzontale
C046 STA $02B0
C049 LDA $D01C
C04C AND $02AE    Multicolor
C04F STA $02AD
C052 LDA $D010
C055 AND $02AE    Sprite oltre 255
C058 STA $02B2
C05B LDA $D01B
C05E AND $02AE    Priorita' sullo sfondo
C061 STA $02B3
C064 LDA #$00
C066 STA $FD
C068 LDA $02A9    Moltiplicando il
C06B CLC          puntatore per 64
C06C ROR A        si ottiene l'inizio
C06D ROR $FD      dell'area dati dello
C06F ROR A        sprite. ($FD/$FE)
C070 ROR $FD
C072 STA $FE
C074 LDY #$00
C076 LDA ($FD),Y
C078 STA $02C0,Y  Salva lo sprite in una
C07B INY          area libera in pag. 2
C07C CPY #$3F
C07E BNE $C076
C080 RTS
```

```
          LOCAZIONI UTILIZZATE
$02A7     Numero dello sprite in gioco.
$02A8     Colore dello sprite.
$02A9     Puntatore allo sprite.
$02AA     Coordinata orizzontale.
$02AB     Coordinata verticale.
$02AC     Espansione verticale.
$02AD     Multicolor.
$02AE     Bit relativo allo sprite e suo
$02AF     Complemento (per AND  OR  EOR)
$02B0     Espansione orizzontale.
$02B1     Da aggiungere a $D000  $D001
$02B2     Posizione sul video oltre 255
$02B3     Priorita' sullo sfondo.
$FD/$FE Puntatore area sprite salvato.
```

```
  TAGLIA SPRITE          SYS xxx

RILOCABILE

C000 LDA #$11     Accende l'emissione
C002 STA $D404    sonora.
C005 LDA #$00     Resetta alcune
C007 STA $F9      locazioni in pag.0
C009 STA $FA      usate come flags.
C00B STA $FB      (interruttori)
C00D LDA $02AA    Coordinata orizzontale
C010 STA $0334    messa in locazioni
C013 STA $0335    libere in pag.3
C016 LDA $02AB    Posizione del taglio
C019 ADC #$09     a seconda che lo
C01B LDX $02AC    sprite sia espanso.
C01E BEQ $C022    oppure no.
C020 ADC #$0A     Detta posizione e'
C022 STA $0336    messa in pag.3 ($0336)
C025 LDA $02B2
C028 BEQ $C02C    $F7 e $F8 =00:
C02A LDA #$01     posizione orizz. < 255
C02C STA $F7      $F7 e $F8 =01:
C02E STA $F8      posizione orizz. > 255
C030 NOP
C031 SEI
C032 LDA $15
C034 STA $0315
C037 CLC          Indirizza l'Interrupt
C038 LDA $14      alla nuova routine
C03A ADC #$AF     in modo rilocabile.
C03C STA $0314
C03F BCC $C044
C041 INC $0315
C044 LDA #$01
C046 STA $D01A    Attiva gli Interrupts
C049 LDA $D011    di Raster.
C04C AND #$7F
C04E STA $D011
C051 CLI
C052 NOP             META' A SINISTRA
C053 LDA $F9      $F9=0 esegue la rout.
C055 BEQ $C059    $F9<128 mezzo sprite
C057 BPL $C071    e' uscito a sinistra.
```

```
C059 LDA $0334   Coord.orizz.mezzo spr.
C05C STA $D401   Modifica il suono.
C05F BNE $C06D   Se $0334 <>0 prosegue.
C061 LDA $F7     E' a 0. Posiz. orizz.
C063 BNE $C06B   > o < di 255?
C065 LDA $F9     Se sono usciti i due
C067 BNE $C0A2   pezzi va alla fine.
C069 INC $F9     $F9 >0 e <128
C06B DEC $F7     Posiz. orizz. < 255
C06D DEC $0334   Mezzo spr. a sinistra.
C070 NOP            META' A DESTRA
C071 LDA $F9     Se il bit7 e' attivato
C073 BMI $C097   salta la routine.
C075 LDY #$FF    Sceglie la fine della
C077 LDA $F8     corsa tra 255 e 96.
C079 BEQ $C07D   Dipende dalla posiz.
C07B LDY #$60    oriz. del mezzo sprite
C07D CPY $0335   Compara con la coord
C080 BNE $C08E   e se diversa prosegue.
C082 CPY #$FF    E' a 255 o a 96?
C084 BEQ $C08C   Se 255 puo' proseguire
C086 LDA $F9     Controlla l'altra meta
C088 BNE $C0A2   Se il flag <>0 esce.
C08A DEC $F9     Setta il bit 7 di $F9
C08C INC $F8     Mezzo sprite oltre 255
C08E INC $0335   Mezzo sprite a destra.
C091 LDA $0335   Modifica la frequenza
C094 STA $D401   della nota (HI)
C097 LDY #$02
C099 JSR $EEB3   Ritardo modificabile
C09C DEY         dal valore del reg. Y
C09D BNE $C099
C09F BEQ $C053   -JMP$C053:ripete ciclo
C0A1 NOP                 FINE
C0A2 JSR $FF8A   Resetta il puntatore
C0A5 LDA #$00    dell'Interrupt.
C0A7 STA $D01A   Disattiva il Raster
C0AA STA $D404   e l'emissione sonora.
C0AD RTS
C0AE NOP         ROUTINE NELL'INTERRUPT
C0AF LDA $D019
C0B2 AND #$01    Se l'Interrupt viene
C0B4 BNE $C0B9   dal Raster prosegue
C0B6 JMP $FEBC   altrimenti esce.
C0B9 STA $D019
C0BC LDA $FA     La routine da eseguire
C0BE BNE $C0ED   e' scelta dal flag $FA
C0C0 LDA $0336
C0C3 STA $D012   Scatto del Raster.
C0C6 LDA $F7     Posizione >o< di 255
C0C8 BEQ $C0D5
C0CA LDA $D010   Modifica il registro
C0CD ORA $02AE   di posizione.
C0D0 STA $D010
C0D3 BNE $C0DE   -JMP $C0DE
C0D5 LDA $D010
C0D8 AND $02AF   Come sopra.
C0DB STA $D010
C0DE LDX $02B1   Va aggiunto a $D000
C0E1 LDA $0334   Coordinata orizzontale
C0E4 STA $D000,X nel reg. appropriato.
C0E7 INC $FA     Setta il flag.
C0E9 JMP $FEBC   Esce.
C0EC NOP
C0ED LDA #$00    Scatto del Raster.
C0EF STA $D012
C0F2 LDA $F8
C0F4 BEQ $C101
C0F6 LDA $D010
C0F9 ORA $02AE
C0FC STA $D010   Questa routine
C0FF BNE $C10A   e' uguale alla
C101 LDA $D010   precedente,
C104 AND $02AF   ma considera
C107 STA $D010   l'altra meta'
C10A LDX $02B1   dello sprite.
C10D LDA $0335
C110 STA $D000,X
C113 DEC $FA
C115 JMP $EA31
```

```
   CAMBIO DEL COLORE        SYS xxx,c

RILOCABILE

C000 JSR $B7F1   Preleva il parametro.
C003 STX $FC     Colore per il cambio.
C005 LDA #$00
C007 STA $FA     Flag nell'Interrupt.
C009 SEI
C00A LDA $15
C00C STA $0315
C00F CLC         Indirizza l'Interrupt
C010 LDA $14     alla nuova routine
C012 ADC #$63    in modo rilocabile.
C014 STA $0314
C017 BCC $C01C
C019 INC $0315
C01C LDA #$81
C01E STA $D01A
C021 LDA $D011   Attiva gli interrupts
C024 AND #$7F    di Raster.
C026 STA $D011
C029 CLI
C02A LDA $02AB   Coordinata verticale.
C02D STA $FB
C02F LDY #$16    Ciclo diverso a
C031 LDA $02AC   seconda che lo sprite
C034 BEQ $C038   sia espanso vertical-
C036 LDY #$2B    mente oppure no.
C038 NOP            INIZIO DEL CICLO
C039 INC $FB     Scatto del Raster.
C03B LDA #$81    Accende l'emissione
C03D STA $D404   sonora.
C040 LDX #$40
C042 JSR $EEB3   Velocita' modificabile
C045 DEX         dal valore del reg. X
C046 BNE $C042
C048 INC $D401   Cambia la nota.
C04B DEY         Decrementa il ciclo.
C04C BNE $C039      FINE DEL CICLO
C04E JSR $FF8A   Resetta il puntatore
```

```
C051 LDA #$00       dell'Interrupt.
C053 STA $D01A      Disattiva il Raster e
C056 STA $D404      l'emissione sonora.
C059 LDX $02A7      Numero dello sprite.
C05C LDA $FC        Assicura il colore del
C05E STA $D027,X    cambio nello sprite.
C061 RTS
C062 NOP            ROUTINE NELL'INTERRUPT
C063 LDA $D019
C066 AND #$01       Se l'Interrupt viene
C068 BNE $C06D      dal Raster prosegue
C06A JMP $FEBC      altrimenti esce.
C06D STA $D019
C070 LDA $FA        La routine da eseguire
C072 BNE $C093      e' scelta dal flag $FA
C074 LDA #$00
C076 STA $D012      Scatto del Raster.
C079 LDA $02AD      Controlla se lo sprite
C07C BEQ $C084      e' multicolor,
C07E EOR $D01C      nel qual caso inverte
C081 STA $D01C      il relativo bit.
C084 LDX $02A7      Numero dello sprite.
C087 LDA $02A8      Colore originale
C08A STA $D027,X    nel registro colore.
C08D INC $FA        Il flag e' settato.
C08F JMP $FEBC      Esce.
C092 NOP
C093 LDA $FB        Allinea il Raster con
C095 STA $D012      la coordinata vertic.
C098 LDA $02AD
C09B BEQ $C0A3      Multicolor
C09D EOR $D01C      (come sopra)
C0A0 STA $D01C
C0A3 LDX $02A7      Numero dello sprite.
C0A6 LDA $FC        Colore del cambio nel
C0A8 STA $D027,X    registro colore.
C0AB DEC $FA        Il flag e' resettato.
C0AD JMP $EA31      Esce.
```

SCHIACCIA SPRITE   SYS xxx

RILOCABILE

```
C000 LDX $02A7      Numero dello sprite.
C003 LDA #$0B       Punta allo sprite
C005 STA $07F8,X    salvato.
C008 LDA #$11       Accende l'emissione
C00A STA $D404      sonora.
C00D LDA #$0A
C00F STA $F9        Ciclo di 10
C011 LDA #$02
C013 STA $FA        Ciclo di 2
C015 LDX #$20
C017 LDY #$1D
C019 LDA #$03
C01B STA $FB        Ciclo di 3
C01D LDA $02C0,X
C020 STA $D401      Copia  bit accesi dei
C024 CPY #$02       bytes indicati da Y
C026 STA $02C0,X    nei bytes central.
C029 DEX            indicati da X e
C02A DEY            cambia la nota ($D401)
C02B DEC $FB
C02D BNE $C01D
C02F LDX #$20
C031 LDY #$23
C033 DEC $FA
C035 BNE $C019
C037 DEC $A9
C039 BNE $C037      Ritardo.
C03B NOP
C03C LDA #$1B
C03E STA $FC        Ciclo di 27
C040 LDA #$1A
C042 STA $AE        Bytes superiori.
C044 LDA #$1D
C046 STA $AF
C048 LDA #$24
C04A STA $F7        Bytes inferiori.
C04C LDA #$21
C04E STA $F8
C050 LDX $AE
C052 LDY $AF        Copia il byte esterno
C054 LDA $02C0,X    nel byte sottostante
C057 STA $02C0,Y    e azzera quello
C05A LDA #$00       esterno.
C05C STA $02C0,X
C05F DEC $AE        Modifica i valori per
C061 DEC $AF        il prossimo giro.
C063 LDX $F7
C065 LDY $F8
C067 LDA $02C0,X
C06A STA $02C0,Y    Come sopra per i
C06D LDA #$00       bytes inferiori.
C06F STA $02C0,X
C072 INC $F7
C074 INC $F8
C076 DEC $FC        Decrementa ciclo 27
C078 BNE $C050
C07A JSR $EEB3
C07D DEC $A9        Ritardo.
C07F BNE $C07A
C081 DEC $F9        Decrementa ciclo 10
C083 BNE $C011
C085 NOP
C086 LDA #$00       Ferma l'emissione
C088 STA $D404      sonora.
C08B STA $02E0      Spegne i bytes centr.
C08E STA $02DE      destro e sinistro.
C091 JSR $EEB3
C094 DEC $A9        Ritardo.
C096 BNE $C091
C098 LDX $02B1      Va aggiunto a $D001
C09B LDA #$FF       Sposta lo sprite
C09D STA $D001,X    fuori lo schermo.
C0A0 LDY #$00
C0A2 LDA ($FD),Y
C0A4 STA $02C0,Y    Rigenera lo sprite
C0A7 INY            salvato.
C0A8 CPY #$3F
C0AA BNE $C0A2
C0AC RTS
```

```
 DIPARTIZIONE DELL'ANIMA    SYS xxx

RILOCABILE

C000 LDA #$11      Accende l'emissione
C002 STA $D404     sonora.
C005 LDA #$00      Resetta il flag
C007 STA $FA       usato nell'Interrupt.
C009 STA $D012     Azzera il Raster.
C00C LDA $02AB     Coordinata verticale.
C00F STA $FB
C011 NOP
C012 SEI
C013 LDA $15
C015 STA $0315
C018 CLC           Indirizza l'interrupt
C019 LDA $14       alla nuova routine
C01B ADC #$4C      in modo rilocabile.
C01D STA $0314
C020 BCC $C025
C022 INC $0315
C025 LDA #$81      Attiva l'Interrupt
C027 STA $D01A     di Raster.
C02A CLI
C02B NOP
C02C LDX #$10
C02E JSR $EEB3     Ritardo modificabile
C031 DEX           dal valore del reg. X
C032 BNE $C02E
C034 LDA $FB       Variazione del suono
C036 EOR #$FF      in crescendo.
C038 STA $D401
C03B DEC $FB       Alza l'anima,
C03D BNE $C02C     finche' arriva a 0
C03F JSR $FF8A     Resetta il puntatore
C042 LDA #$00      dell'Interrupt.
C044 STA $D01A     Disattiva il Raster
C047 STA $D404     e l'emissione sonora.
C04A RTS
C04B NOP           ROUTINE NELL'INTERRUPT
C04C LDA $D019
C04F AND #$01      Se l'Interrupt viene
C051 BNE $C056     dal Raster prosegue
C053 JMP $FEBC     altrimenti esce.
C056 STA $D019
C059 LDA $FA       La routine da eseguire
C05B BNE $C073     e' scelta dal flag $FA
C05D LDY $02B1     Va aggiunto a $D001
C060 LDA $FB       Coordinata verticale
C062 STA $D001,Y   dell'anima.
C065 LDY $02A7     Numero dello sprite.
C068 LDA #$01      Colore bianco (anima)
C06A STA $D027,Y   nel registro colore.
C06D INC $FA       Il flag e' settato.
C06F JMP $EA31     Esce.
C072 NOP
C073 LDY $02B1     Va aggiunto a $D001
C076 LDA $02AB     Coordinata dello
C079 STA $D001,Y   sprite fermo.
C07C LDY $02A7     Numero dello sprite.
C07F LDA $02A8     Colore originale
C082 STA $D027,Y   nel registro colore.
C085 DEC $FA       Il flag e' resettato.
C087 JMP $EA31     Esce.
```

```
 DISINTEGRAZIONE          SYS xxx

RILOCABILE

C000 LDA #$11      Accende l'emissione
C002 STA $D404     sonora.
C005 LDA #$00      Resetta due locazioni
C007 STA $FA       in pag.0
C009 STA $FC       usate nella routine.
C00B NOP             INIZIO DEL CICLO
C00C LDY #$3F      Numero dei bytes di
C00E INC $FC       uno sprite.
C010 INC $FC
C012 DEC $FA
C014 DEC $FA
C016 DEY           prende i bytes dallo
C017 LDA $02C0,Y   sprite salvato, ne
C01A AND $FA       modifica i bits e li
C01C AND $FB       stampa nello sprite
C01E STA ($FD),Y   considerato.
C020 STA $D401     Suono casuale.
C023 INC $FB
C025 LDX $FC
C027 DEX           Ritardo crescente.
C028 BNE $C027
C02A TYA
C02B BNE $C016       FINE DEL CICLO
C02D LDA $FA       Ricomincia finche'
C02F BNE $C00C     $FA=0.
C031 LDA #$10
C033 STA $D404     Spegne la forma d'onda
C036 RTS
```

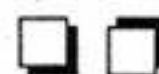
ENCICLOPEDIA DI ROUTINES

Per avvicinarsi ad un nuovo linguaggio non c'è nulla di meglio che confrontare brevi programmi che svolgono le stesse funzioni

Come anticipato (ed, in parte, già attuato da un po' di tempo) abbiamo preso la decisione, a partire da questo numero, di pubblicare sistematicamente argomenti relativi ai più diffusi linguaggi di programmazione oggi disponibili sui computer moderni; con queste parole (non si dispiacciano gli irriducibili **64**isti) intendiamo prendere in considerazione **esclusivamente** gli elaboratori della fascia **Ms - Dos** ed **Amiga**.

I programmi

Nelle pagine dedicate all'enciclopedia trovere due "tipi" di spiegazioni, entrambe relative ad uno stesso algoritmo. Ognuno di questi, infatti, "gira" allo stesso modo, qualunque sia il linguaggio nel quale è stato implementato; del resto, non potrebbe esser diversamente, dal momento che uno degli scopi dell'enciclopedia è quello di far confrontare tra loro le procedure, da seguire nei vari linguaggi, per giungere al medesimo risultato.

A parte, quindi, verranno evidenziate le note esplicative sulle tecniche e sulle differenze usate nell'implementazione dei tre linguaggi principali.

Una barra proporzionale

I programmi consentono di visualizzare, in modalità grafica, una barra colorata (orizzontale o verticale) che, partendo da una lunghezza nulla, riempe un po' per

Perchè un'enciclopedia

Molto, molto tempo fa veniva pubblicata, in queste stesse pagine, una rubrica che riscosse un notevole successo presso gli smanettoni; si trattava dell'enciclopedia di routine in **Basic** cui si affiancò, di lì a poco, l'enciclopedia di routine in **linguaggio macchina**.

Nel primo caso si trattava, in effetti, di subroutines (ovviamente in Basic) contraddistinte dalla notevole "universalità", abbastanza versatili da essere inserite in programmi di più ampio resipro. L'enciclopedia di routine in l.m., invece, comprendeva listati interamente rilocabili e, per questo motivo, offriva potenzialità insperate anche ai possessori dei piccoli **C/64**.

Con il passar del tempo, però, ci si rese conto che l'accodamento di tutte le routine superava di gran lunga la modesta memoria del C/64; la possibilità di effettuare operazioni di **merge** (=mescolamento di brani di programma) presentava le sue brave difficoltà e, spesso, alcune routine potevano presentare problemi di vario tipo se utilizzate insieme.

Il motivo delle crescenti difficoltà era dovuto all'impossibilità di usare **variabili locali** ed alla modesta quantità di memoria Ram a disposizione (una schermata grafica rubava spazio ai programmi e viceversa). Per farla breve, la rubrica fu sospesa in attesa di tempi migliori.

Questi, ora, sono finalmente arrivati, grazie soprattutto alle sostanziali migliorie offerte (a livello di H/w e S/w) dai **moderni computer**.

Inoltre, proprio per la proliferazione di linguaggi di notevole affidabilità, è finalmente possibile implementare uno stesso algoritmo servendosi di linguaggi diversi. Come il lettore avrà modo di osservare, infatti, sia la "struttura" di un programma, sia la stessa sintassi, offrono somiglianze incredibili, insospettabili fino a qualche tempo fa.

La "nuova" edizione dell'enciclopedia, pertanto, rappresenta un'occasione da non perdere per chi intende avanzare con sicurezza nel meraviglioso mondo della programmazione.

volta un rettangolo vuoto. La routine non si limita, quindi, a disegnare un rettangolo colorato (operazione, del resto, semplicissima e già disponibile immediatamente in tutti i linguaggi), ma consente di render visibile un conteggio, in modo analogico, durante l'elaborazione di un qualunque programma principale.

Il lettore avrà sicuramente osservato numerosi programmi professionali (di solito utility) che, per per far intuire all'utilizzatore il tempo necessario al compimento di una certa operazione, tracciano un rettangolo che si allunga, in proporzione al tempo trascorso, fino a riempire un rettangolo vuoto; quest'ultimo rappresenta il tempo richiesto per il completamento dell'operazione.

Di solito il tipo di visualizzazione descritto viene usato durante le copie di files o l'invio di dati verso la stampante (o verso un modem), operazioni notoriamente lunghe.

Alla prima domanda che compare (**Valore massimo del range**), non appena il programma parte, bisogna rispondere indicando il valore massimo che, nella nostra simulazione, l'ipotetico programma principale deve raggiungere. Supponiamo, ad esempio, di rispondere con 5000. La **seconda** domanda impone una scelta sul **posizionamento** della barra stessa; questa occuperà la parte superiore dello schermo (0= **orizz.**) oppure la parte sinistra del video (1= **vert.**).

La routine consente anche di non far occupare l'intero spazio a disposizione sul video. Se volete, infatti, che la barra occupi soltanto metà della lunghezza a disposizione, digiterete **50** in risposta alla **terza** domanda; se volete l'intera occupazione dello schermo, risponderete **100**; e così via.

La **quarta** domanda, onde evitare di complicare notevolmente il programma, richiede la digitazione del valore, espresso in pixel, delle dimesioni del video. A seconda della scheda grafica in vostro possesso, quindi, digiterete i due valori richiesti, in successione. In effetti si richiede la digitazione dei due valori relativi ad una "**finestra**", le cui dimensioni verrano prese come riferimento per posizionare la barra al suo interno. All'inizio, comunque, inserite i valori relativi all'intera scheda video (esempio, 640, 200 nel caso di **Vga** o similari); in seguito proverete ad inserire valori diversi per vedere che cosa succede.

```
 ' Versione QuickBasic (AT compatibile)
 DECLARE SUB DisegnaUnaBarra (InPosizione, Percentuale, DiColo-
rel$, Su, Colore2$, DiValore)
 DIM SHARED PixOriz, PixVert, y
 ' Con ValMax=5000, 100% PixVert=639, PixOr=199 Tempo= 10 secondi
 DiColorel$ = "Blu": Colore2$ = "Rosso": CLS
 INPUT "Valore max. range"; ValoreMax: ValoreMax = ABS(ValoreMax)
 InPosizione$ = "ORIZ": INPUT "Posizione barra (0 = Orizz. 1=
Vert)"; InPosizione: IF InPosizione <> 0 THEN InPosizione$ = "vert"
 INPUT "Valore % occupazione schermo (>0 / 100)"; Percentuale:
Percentuale = ABS(Percentuale): IF Percentuale > 100 OR Percentuale
= 0 THEN Percentuale = 100
 INPUT "N.Pixel Orizz. schermo"; PixOriz
 INPUT "N.Pixel Vert. schermo"; PixVert
 IF UCASE$(InPosizione$) = "ORIZ" THEN
 InPosizione = 1: DiLunghezza = ValoreMax / PixOriz
 ELSE : InPosizione = 0: DiLunghezza = ValoreMax / PixVert
 END IF
 CLS : SCREEN 9: LOCATE 19, 20: PRINT "Valore massimo ="; ValoreMax
 LOCATE 20, 20: PRINT "Ogni pixel vale ="; DiLunghezza
 LOCATE 21, 20: PRINT "Valore attuale: "
 ' PROGRAMMA DIMOSTRATIVO
 LOCATE 10, 10: PRINT "Ora inizio test: "; TIME$
 FOR y = 1 TO ValoreMax STEP DiLunghezza: DiValore = y / DiLunghezza
 ' InPosizione (0 vert. / 1 oriz.). Percentuale (occup. schermo)
 ' DiColorel$ (tinta cornice). DiColore2$ (tinta barra)
 ' DiValore (lung. barra da colorare all'interno della cornice)
 DisegnaUnaBarra InPosizione, Percentuale, DiColorel$, Su, Colo-
re2$, DiValore
 NEXT: LOCATE 11, 10: PRINT "Ora fine Test: "; TIME$

 SUB DisegnaUnaBarra (InPosizione, Percentuale, DiColorel$, Su,
Colore2$, DiValore)
 SELECT CASE UCASE$(DiColorel$)
 CASE IS = "ROSSO"
 col1 = 4
 CASE IS = "BLU"
 col1 = 9
 CASE ELSE
 col1 = 3
 END SELECT
 SELECT CASE UCASE$(Colore2$)
 CASE IS = "ROSSO"
 col2 = 4
 CASE IS = "BLU"
 col2 = 9
 CASE ELSE
 col2 = 3
 END SELECT
 IF col1 = col2 THEN col2 = col2 + 1
 IF Percentuale = 0 OR Percentuale > 100 THEN Percentuale = 100
 LOCATE 21, 36: PRINT y
 REM BARRA VERTICALE
 IF InPosizione = 0 THEN
 LINE (1, 1)-(40,INT((PixVert / 100)*Percentuale - 1)), col1, B
 LINE (2, 2)-(39,INT((DiValore / 100)*Percentuale - 1)), col2, BF
 ELSEIF InPosizione = 1 THEN
 REM BARRA ORIZZONTALE
 LINE (1,1)-(INT((PixOriz / 100)*Percentuale), 20), col1, B
 LINE (2,2)-(INT((DiValore / 100)*Percentuale - 1), 19), col2, BF
 END IF
 END SUB
```

La versione Quick Basic di "Disegna una barra"

Dopo aver battuto l'ultimo Return, lo schermo verrà cancellato ed una barra rossa verrà tracciata all'interno di un rettangolo blu, fino a riempirlo.

Alcune indicazioni compaiono al termine della procedura; si tratta di un pro-memoria in cui si ricorda il valore inserito (qualunque sia questo, infatti, la barra riempirà sempre per intero il rettangolo) e del valore corrispondente ad ogni pixel visualizzato; durante l'elaborazione, inoltre, compare costantemente, in tempo reale, il valore "trattato" in quel particolare momento.

Della procedura, pubblichiamo la versione in Quick Basic e in Turbo C.

Girandola

Anche questo programma visualizza... il trascorrer del tempo, anche se in una forma piuttosto rozza e semplicistica.

Il vantaggio, rispetto alla procedura prima descritta, risiede nel fatto che non è necessario il ricorso allo **schermo grafico**, ma tutto si svolge in modalità testo.

Chi ha avuto modo di osservare il programma compattatore **Pkzip** avrà notato che, durante il suo funzionamento, utilizza alcuni caratteri che, visualizzati in successione nella stessa cella video, danno l'impressione di una girandola che gira. Questo particolare consente di far capire, all'utente, che una certa (lunga) operazione è in corso e che l'apparente staticità del video (o la mancata risposta del computer alla pressione dei tasti) non indica necessariamente... l'avvenuto inchiodamento del computer.

I programmi pubblicati, quindi, consentono di visualizzare tre "eliche" affiancate in orizzontale. L'elica a destra, non appena compie un giro completo, fa "scattare" di una posizione quella centrale; questa, quando a sua volta compie un giro completo, fa scattare di una posizione quella posta a sinistra.

In pratica, si tratta di un contatore a tre cifre, pur se operante in modo inconsueto.

Durante il conteggio compare anche il tempo trascorso dal momento del Run; sarà interessante, per il lettore, notare la differenza esistente tra i vari linguaggi ed

```
/* BARRA.C      Versione TurboC 2.0 */
/* Adattamento by Mariani G.        */

/* Include files */
#include <stdio.h>
#include <conio.h>
#include <graphics.h>
#include <string.h>
#include <stdlib.h>
#include <dos.h>

/* Variabuli globali */
float DiValore;
float yf, PixOriz, PixVert, ValoreMax, Percentuale, DiLunghezza;
char DiColore1[10], Colore2[10], InPosizione[10];
int g_driver, g_mode,k;          /* Variabili per tipo grafica  */
int y, InPosiz;
struct time ora;

/* Prototipo di funzione DisegnaUnaBarra */
void DisegnaUnaBarra (int InPosiz, float Percentuale,
         char *DiColore1, char *Colore2, float DiValore, int y);


/* Procedura DisegnaUnaBarra */
void DisegnaUnaBarra (int InPosiz, float Percentuale,
          char *DiColore1, char *Colore2, float DiValore, int y)
{
  int col1, col2;

  strupr(DiColore1);
  col1=3;
  if (strcmp(DiColore1,"ROSSO")==0) col1=4;
  if (strcmp(DiColore1,"BLU")==0) col1=9;

  strupr(Colore2);
  col2=3;
  if (strcmp(Colore2,"ROSSO")==0) col2=4;
  if (strcmp(Colore2,"BLU")==0) col2=9;

  if (col1==col2) col2++;
  if (Percentuale<=0 || Percentuale>100) Percentuale=100;

  gotoxy(38,21); printf("%d",y);

/* BARRA VERTICALE */
  if (InPosiz==0)
  {
    setfillstyle(LTSLASH_FILL,col1);
    bar(1,1,40,(PixVert/100)*Percentuale-1);
    setfillstyle(SOLID_FILL,col2);
    bar(2,2,39,(DiValore/100)*Percentuale-1);
  }
/* BARRA ORIZZONTALE */
  if (InPosiz==1)
  {
    setfillstyle(LTSLASH_FILL,col1);
    bar(1,1,(PixOriz/100)*Percentuale,20);
    setfillstyle(SOLID_FILL,col2);
    bar(2,2,(DiValore/100)*Percentuale,19);
  }
}
```

Versione Turbo C Borland (Disegna barra); prima parte

a seconda se si possiede un microprocessore matematico oppure no.

Barra Qick Basic

Da notare l'utilizzo di variabili dal nome lunghissimo (attenzione alla loro corretta digitazione!) che presentano il (limitato?) vantaggio di rendere molto semplice la comprensione del funzionamento delle routines.

Le variabili **PixOriz**, **PixVert** ed **Y** sono dichiarate di tipo *Shared*, in modo da renderle "visibili" all'intero programma. Il valore di 10 secondi (per ciò che riguarda il tempo del test) viene riferito a computer AT (10 Mhz) dotato di processore matematico.

Le variabili **Dicolore1$** e **Colore2$** possono esser diversamente definite dall'utente, a patto di tener conto della differenza di valori anche nella subroutine successiva.

Per ciò che riguarda la stringa **InPosizione$**, la sintassi seguita è certamente ridondante: al lettore il compito di snellirla, una volta compreso lo scopo.

L'uso di **Select** è usato per ricordare al lettore principiante l'alternativa offerta, da Quick Basic, per effettuare "salti" diversi dal solito If... Then. **Ucase$**, invece, consente di evitare errori anche nel caso in cui si digitasse una stringa senza tener conto dei caratteri maiuscoli o minuscoli. Il comando grafico **Line** contiene alcuni parametri elaborati dalla routine; oltre a questo, però, non ha nulla di particolare.

Si consiglia di studiare in modo approfondito (sfogliando il manuale della **confezione originale Microsoft**) la sintassi If... Then... Endif e tutte le altre forme sintattiche "insolite" per i neo utenti dei sistemi **Ms - Dos**.

Si noti, infine, la presenza delle istruzioni relative al conteggio del tempo trascorso; potranno essere utili sia ai lettori che intendano far girare il programma su computer diversi, sia a coloro che intendano apportare modifiche atte a velocizzare la procedura.

```
 void main()
 {
   strcpy(DiColore1,"BLU");
   strcpy(Colore2,"ROSSO");
   clrscr();
   printf("Valore massimo del range : ");
   scanf("%f",&ValoreMax); ValoreMax=abs(ValoreMax);


   strcpy(InPosizione,"ORIZ");
   printf("Posizione barra (0 = Orizz. 1 = Vert) : ");
   scanf("%d",&InPosiz);
          if (InPosiz!=0) strcpy(InPosizione,"VERT");


   printf("Valore %% occupazione schermo (>0 / 100) : ");
   scanf("%f",&Percentuale); Percentuale=abs(Percentuale);
   if (Percentuale>100 || Percentuale<=0) Percentuale=100;


   printf("N.Pixel Oriz. schermo : "); scanf("%f",&PixOriz);
   printf("N.Pixel Vert. schermo : "); scanf("%f",&PixVert);


   if (strcmp(InPosizione,"ORIZ")==0)
     { InPosiz=1; DiLunghezza=ValoreMax/PixOriz; }
   else
     { InPosiz=0; DiLunghezza=ValoreMax/PixVert; }


detectgraph(&g_driver, &g_mode);
/* Determina la scheda grafica presente */
/* Schermo posto in grafica. Notare: "c:\tc\bgi"*/
initgraph(&g_driver, &g_mode, "c:\\tc\\bgi");

cleardevice();
  gotoxy(20,19); printf("Valore massimo   : %f",ValoreMax);
  gotoxy(20,20); printf("Ogni pixel vale  : %f",DiLunghezza);
  gotoxy(20,21); printf("Valore attuale   : ");

/* Programma dimostrativo */
  gotoxy(10,10); printf("Ora inizio test : ");
  gettime(&ora);
  printf("%d:%d:%d",ora.ti_hour,ora.ti_min,ora.ti_sec);
  for (y=1; y<=ValoreMax; y=y+DiLunghezza)
  {
    yf=y; /* Converte y in float */
    DiValore=y/DiLunghezza;

/* InPosiz (0 verticale / 1 orizzontale).               */
/* Percentuale (occupazione schermo)                     */
/* DiColore1 (tinta cornice). Colore2 (tinta barra) */
/* DiValore (lunghezza della barra da colorare          */
/* all'interno della cornice)*/

DisegnaUnaBarra (InPosiz, Percentuale, DiColore1,
Colore2, DiValore, y);
  }
  gotoxy(10,11); printf("Ora fine test    : ");
  gettime(&ora);
  printf("%d:%d:%d",ora.ti_hour,ora.ti_min,ora.ti_sec);
}
```

Versione Turbo C Borland (Disegna barra); parte finale

Barra C

La procedura **DisegnaUnaBarra** che, come dice il nome, disegna la barra sul video, possiede alcuni parametri che sono: **InPosiz**; se vale 1 indica che la barra deve essere orizzontale, se è 0 deve essere verticale. **Percentuale**; indica la percentuale di schermo occupata. **DiColore1**, **Colore2**; sono i colori del bordo e dell'interno della barra.

DiValore è il valore relativo della barra da disegnare; **y** è il valore assoluto della barra da disegnare.

Per ciò che riguarda le istruzioni della procedura "**DisegnaUnaBarra**" si precisa che: **strupr** trasforma in caratteri maiuscoli la stringa specificata; **strcmp** esegue il confronto tra la stringa ed una sottostringa specificata: restituisce **0** se le due stringhe sono uguali, altrimenti un valore positivo o negativo a seconda che la stringa sia maggiore o minore della sottostringa specificata.

Il comando **gotoxy**, come in T.P., posiziona il cursore alle coordinate specificate.

Il comando **setfillstyle** decide il "disegno" ed il colore con il quale deve essere riempita la figura che verrà disegnata in seguito. LTSLASH_FILL significa riempimento con barre oblique, SOLID_FILL un riempimento totale; **bar** disegna un rettangolo.

Il programma principale (**Main program**) chiede i vari parametri in input e richiama la procedura "**disegnaunabarra**". Nel Main sono degni di nota: **clrscr** (cancella lo schermo); **strcpy** (assegna la stringa specificata con il valore specificato; **detectgraph** (determina automaticamente il tipo di scheda grafica presente); **initgraph** (pone lo schermo in modo grafico). Si noti che il parametro "**c:\\tc\\bgi**" indica la directory in cui devono essere presenti i driver grafici del C, cioè tutti quelli con suffisso "*.BGI". Ricordiamo che questi files devono essere presenti anche nel caso di programma compilato ".EXE".

Il comando **cleardevice** cancella lo schermo grafico, mentre **gettime** legge l'orologio di sistema.

```
' Versione QuickBasic (AT compatibile. Tempo test= 18 secondi)
DECLARE SUB Visualizza (Immagine)
DIM SHARED Immagine
Immagine = 1: Centinaia = 1: Migliaia = 1: CLS
LOCATE 10, 7: Immagine = Migliaia: Visualizza Immagine
LOCATE 10, 8: Immagine = Centinaia: Visualizza Immagine
LOCATE 10, 9: Visualizza Immagine
LOCATE 1, 1: PRINT "Ora di inizio: "; TIME$
FOR Conteggio = 1 TO 5000
LOCATE 8, 1: PRINT "Conteggio: "; Conteggio
IF INT(Conteggio / 1000) = Conteggio / 1000 THEN
        LOCATE 10, 7: Migliaia = Migliaia + 1
                IF Migliaia > 4 THEN Migliaia = 1
        Immagine = Migliaia: Visualizza Immagine
END IF
IF INT(Conteggio / 100) = Conteggio / 100 THEN
        LOCATE 10, 8: Centinaia = Centinaia + 1
                IF Centinaia > 4 THEN Centinaia = 1
        Immagine = Centinaia: Visualizza Immagine
END IF
LOCATE 10, 9: Visualizza Immagine: Immagine = Immagine + 1
                IF Immagine > 4 THEN Immagine = 1
FOR Ritardo = 1 TO 100: NEXT Ritardo: REM eventuale ritardo
NEXT Conteggio: LOCATE 2, 1: PRINT "Ora di fine test: "; TIME$

SUB Visualizza (Immagine)
SELECT CASE Immagine
CASE IS = 1
PRINT CHR$(179); : ' Verticale
CASE IS = 2
PRINT CHR$(47); : ' Slash
CASE IS = 3
PRINT CHR$(196); : ' Orizzontale
CASE IS = 4
PRINT CHR$(92); : ' Back Slash
CASE ELSE
PRINT ;
END SELECT
END SUB
```

"Giranfola"; versione Quick Basic Microsoft

```
/* GIRANDOLA.C      Versione TurboC 2.0   */
/* Adattamento by Mariani G.               */


/* Include files */
#include <stdio.h>
#include <conio.h>
#include <dos.h>


/* Dichiarazione variabili globali */
int Immagine, Centinaia, Migliaia, Conteggio, Ritardo;
struct time ora;


void Visualizza (int Immagine);
/* Prototipo funzione Visualizza */
```

"Girandola"; versione Turbo C Borland (prima parte)

Girandola Quick Basic

C'è poco da dire, in questo listato, che non si discosti molto da ciò che è stato detto per il listato precedente.

L'unica nota sufficientemente importante è relativa alla visualizzazione dei caratteri che generano la "girandola". Si tratta di quattro innocui caratteri semigrafici il cui codice Ascii è, rispettivamente, **179** (barra verticale), **47** (inclinazione a destra), **196** (barra orizzontale), **92** (inclinazione a sinistra). La successione delle "immagini" provoca l'illusione ottica del movimento della girandola.

Girandola T. Pascal

Da notare, rispetto alla versione Q. Basic, il notevole incremento di velocità di elaborazione.

Inutile soffermarsi sulla necessità di dichirare alcune clausole iniziali standard (**uses**...): maggiori informazioni si possono rintracciare sul **manuale accluso alla confezione originale del Turbo Pascal Borland**.

E' bene, invece, sottolineare la notevole somiglianza esistente tra il comando **Case of** disponibile sia in Q.B. (pur se con sintassi lievemente diversa) che in T.P.

Da notare, soprattutto, la necessità di digitare nella parte **iniziale** del listato la **procedura** (e, se ve ne fossero, eventuali funzioni) anzichè in "coda" al listato stesso.

L'eccessiva velocità di alcuni computer (basati sul micro **80486**) potrebbe suggerire la necessità di inserire la linea di ritardo "occultata", verso la fine del programma pubblicato, da una provvidenziale Rem.

L'opportuna modifica dei parametri dell'istruzione **Goto** (incaricata di "posizionare" il cursore in una zona ben precisa del video) consentirà di far apparire i tre caratteri semigrafici in una posizione qualunque dello schermo.

```
void Visualizza (int Immagine)
{
  char a;

  switch (Immagine)
  {
    case 1: a=179;    /* Verticale    */
            break;
    case 2: a=47;     /* Slash        */
            break;
    case 3: a=196;    /* Orizzontale */
            break;
    case 4: a=92;     /* BackSlash    */
            break;
    default: a=32;
             break;
  }
  printf("%c",a);
}

void main()
{
  Immagine=1;
  Centinaia=1;
  Migliaia=1;

  clrscr();  /* Cancella lo schermo */

  gotoxy(7,10); Immagine=Migliaia; Visualizza (Immagine);
  gotoxy(8,10); Immagine=Centinaia; Visualizza (Immagine);
  gotoxy(9,10); Visualizza (Immagine);

  gotoxy(1,1);
  printf("Ora di inizio test : ");
  gettime(&ora);
  printf("%d:%d:%d",ora.ti_hour,ora.ti_min,ora.ti_sec);

  for (Conteggio=1; Conteggio<=2000; Conteggio++)
  {
    gotoxy(1,8); printf("Conteggio : %d",Conteggio);
    if ((Conteggio % 1000) == 0)
    {
      gotoxy(7,10); Migliaia++;
      if (Migliaia>4) Migliaia=1;
      Immagine=Migliaia; Visualizza(Immagine);
    }
    if ((Conteggio % 100) == 0)
    {
      gotoxy(8,10); Centinaia++;
      if (Centinaia>4) Centinaia=1;
      Immagine=Centinaia; Visualizza(Immagine);
    }
    gotoxy(9,10); Visualizza(Immagine); Immagine++;
    if (Immagine>4) Immagine=1;
    for (Ritardo=1; Ritardo<3000; Ritardo++);
/* Eventuale ritardo */
  }
  gotoxy(1,2); printf("Ora di fine test   : ");
  gettime(&ora);
  printf("%d:%d:%d",ora.ti_hour,ora.ti_min,ora.ti_sec);
}
```

"Girandola" versione Turbo C Borland (continuazione e fine)

Girandola C

La procedura "**Visualizza**", ovviamente, visualizza il carattere semigrafico che simula la rotazione della girandola. Il parametro "**Immagine**" indica il carattere da stampare.

La procedura **switch... case** permette di eseguire alcune istruzioni, piuttosto che altre, a seconda del valore della variabile specificata in **switch**; **break**, invece, consente l'uscita dal ciclo **switch... case**.

Il "**Main Program**" esegue il ciclo principale, richiamando la funzione di visualizzazione del carattere.

I listati in C

I due listati in C presentano due parti abbastanza simili, sulle quali ci soffermiamo per andare incontro alle esigenze dei principianti.

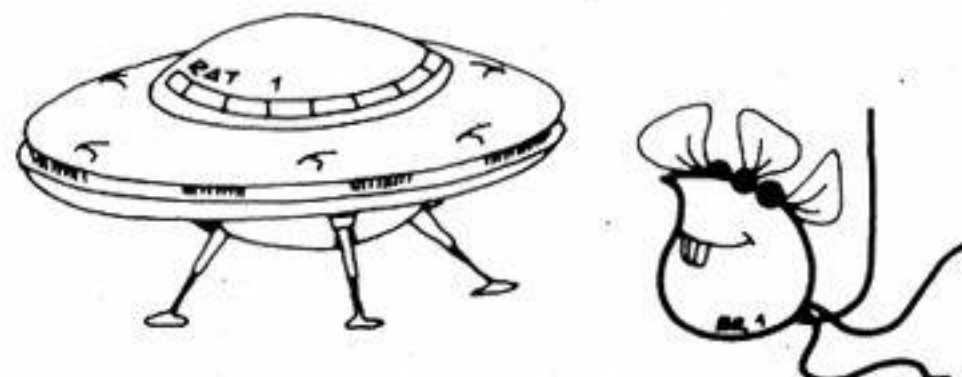
Include files. I files di include (istruzioni **#include <nome>**) contengono tutte le dichiarazioni necessarie al corretto funzionamento delle istruzioni C.

Le variabili **globali** sono variabili che vengono "viste", e quindi possono essere usate, dall'**intero** programma C, subroutine e funzioni comprese. Ricordiamo che in C (come in T.P.) tutte le variabili usate dal programma devono obbligatoriamente essere dichiarate, pena segnalazione di errore da parte del compilatore.

Le variabili devono essere dichiarate anteponendo, al nome della variabile, il **TIPO** della stessa. In **C** possono esserci variabili intere (**int**), in virgola mobile (**void**), carattere o stringa (**char**), ecc.

I **prototipi di funzione** eseguono la stessa funzione dei files di tipo "**.H**"; in questo caso, però, non per le istruzioni standard C, ma per le procedure e funzioni definite dall'utente. I prototipi dicono al compilatore di che tipo è la procedura

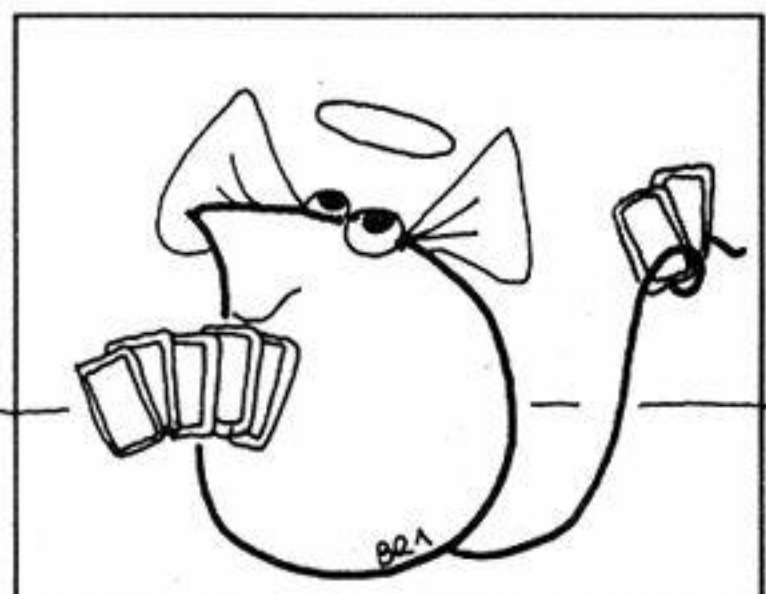
```
(* Versione TurboPascal AT comp. Tempo test= 11 secondi ca.*)
uses crt, dos;
var Centinaia, Migliaia, Conteggio, Immagine, Ritardo: integer;
var h, m, s, d: word;
procedure Visualizza (Immagine:integer);
begin
CASE Immagine of
1: writeln (CHR (179)); (*Verticale*)
2: writeln (CHR (47)); (*Slash*)
3: writeln (CHR (196)); (*Orizzontale*)
4: writeln (CHR (92)); (*Back Slash*)
END;
end;
begin;clrscr;
Migliaia :=1;Centinaia:=1;
GOTOXY(7, 10);
Immagine := Migliaia; Visualizza (Immagine);
gotoXY(8,10);Immagine := Centinaia; Visualizza (Immagine);
Immagine:=1;
GoToXY(9, 10);Visualizza (Immagine);
GoToXY(1, 1);settime(0,0,0,0);gettime(h,m,s,d);
writeln ('Ora di inizio: ',h,m,s,d);
for Conteggio :=1 to 5000
    do
      begin;
      GoToXY(1, 8);Writeln('Conteggio: ',conteggio);
              IF round(Conteggio / 1000) = Conteggio / 1000 THEN
                  begin;
                      GoToXY(7, 10);Migliaia := Migliaia + 1;
                               IF Migliaia > 4 THEN
                                       begin;
                                             Migliaia := 1;
                                       end;
                   Immagine := Migliaia; Visualizza (Immagine);
                  end;
                IF round(Conteggio / 100) = Conteggio / 100 THEN
                    begin;
                      GoToXY(8, 10);Centinaia := Centinaia + 1;
                                       IF Centinaia > 4 THEN
                                          begin;
                                                Centinaia := 1;
                                          end;
                  Immagine := Centinaia; Visualizza (Immagine);
                  end;
GoToXY(9, 10);Visualizza (Immagine);Immagine := Immagine + 1;
                IF Immagine > 4 THEN Immagine := 1;
(*FOR Ritardo := 1 TO 299 do begin; end;*)
end;
GoToXY(1, 2);gettime(h,m,s,d);
writeln('Ora di fine test: ',h,m,s,d);
repeat until keypressed;
end.
```

Il listato "Girandola" versione Turbo Pascal

e quanti e quali parametri possiede; ciò allo scopo di permettere al compilatore stesso di effettuare i dovuti controlli per evitare il passaggio di parametri errati.

In C possono esserci **Procedure** o **Funzioni**. Le procedure eseguono un determinato compito **senza** restituire alcun valore all'istruzione chiamante. Per questo tipo la dichiarazione deve essere preceduta dalla parola "void", che significa, appunto, che la procedura non ha parametri di ritorno. Una "funzione", invece, restituisce un parametro. Per dichiarare una funzione, al posto della parola "void" bisogna inserire uno dei tipi ammessi in C (int, float, ecc.) in modo da definire il tipo della funzione dichiarata.

Un esempio di procedura è **printf**, che non restituisce parametri. Un esempio di funzione, invece, è **eof**, che restituisce la condizione di fine del file.

Nella fase di scrittura di eventuali procedure o funzioni andranno scritte le istruzioni componenti le eventuali procedure o funzioni presenti nel programma C. Ognuna di queste può utilizzare le variabili globali dichiarate in testa al programma, oppure avere proprie variabili locali dichiarate all'interno della procedura.

Il **main** è la parte di programma che verrà eseguita all'atto del richiamo del programma C, e che quindi deve contenere tutte le istruzioni necessarie alle operazioni da svolgere ed al richiamo di eventuali procedure o funzioni presenti nel programma. Il main si dichiara alla pari di una qualsiasi altra procedura o funzione, appunto di nome "main". Può anche ricevere parametri dall'esterno (in questo caso, dal DOS), e restituire parametri (sempre al DOS).

```
REM Versione Amiga
' Con ValMax=5000, 100% orizz, PixOr=639, PixVert=199 Tempotest
= 43 secondi

DIM SHARED PixOriz, PixVert, y
DiColore1$ = "Nero": Colore2$ = "Bianco"

INPUT "Valore massimo del range"; ValoreMax
ValoreMax = ABS(ValoreMax)
InPosizione$ = "ORIZ"

INPUT "Posizione barra (0 = Orizz. 1= Vert)"; InPosizione
IF InPosizione <> 0 THEN InPosizione$ = "vert"

INPUT "Valore % occupazione schermo (>0 / 100)"; Percentuale
Percentuale = ABS(Percentuale)
IF Percentuale > 100 OR Percentuale = 0 THEN Percentuale = 100

INPUT "N.Pixel Orizz. schermo (639)"; PixOriz

INPUT "N.Pixel Vert. schermo (199)"; PixVert
IF UCASE$(InPosizione$) = "ORIZ" THEN
InPosizione = 1: DiLunghezza = ValoreMax / PixOriz
ELSE : InPosizione = 0: DiLunghezza = ValoreMax / PixVert
END IF
CLS

LOCATE 19,20: PRINT "Valore massimo  ="; ValoreMax
LOCATE 20,20: PRINT "Ogni pixel vale ="; DiLunghezza
LOCATE 21,20: PRINT "Valore attuale:"

' PROGRAMMA DIMOSTRATIVO
LOCATE 10,10:PRINT "Ora inizio Test: "TIME$
FOR y = 1 TO ValoreMax STEP DiLunghezza: DiValore = y / DiLunghezza
DisegnaUnaBarra InPosizione, Percentuale, DiColore1$, Su, Colo-
re2$, DiValore
NEXT:LOCATE 11,10:PRINT "Ora fine Test:  "TIME$:END
SUB DisegnaUnaBarra (InPosizione, Percentuale, DiColore1$, Su,
Colore2$, DiValore) STATIC
IF UCASE$(DiColore1$)="NERO" THEN col1 =1
IF UCASE$(DiColore1$)="BIANCO" THEN col1 = 2
IF UCASE$(Colore2$)="NERO" THEN col2 = 1
IF UCASE$(Colore2$)="BIANCO" THEN col2 = 2
IF col1 = col2 THEN col2 = col2 + 1
IF Percentuale = 0 OR Percentuale > 100 THEN Percentuale = 100
LOCATE 21, 36: PRINT y
REM BARRA VERTICALE
IF InPosizione = 0 THEN
LINE (1,1)-(40, INT((PixVert / 100) * Percentuale - 1)), col1, B
LINE (2,2)-(39, INT((DiValore / 100)*Percentuale - 1)), col2, BF
ELSEIF InPosizione = 1 THEN
REM BARRA ORIZZONTALE
LINE (1,1)-(INT((PixOriz / 100) * Percentuale), 20), col1, B
LINE (2,2)-(INT((DiValore / 100)*Percentuale - 1), 19), col2, BF
END IF
END SUB
```

"Disegna una barra"; versione AmigaBasic

Si notino le minime differenze tra la versione **AmigaBasic** e quella in **Quick Basic Microsoft**. Tali trascurabili diversità ci hanno indotto a non pubblicare anche la versione AmigaBasic del programma *"Girandola"*. Siamo sicuri che il lettore, da solo, sia in grado di apportare le correzioni del caso, basandosi esclusivamente sulla lettura della versione Q. Basic.

di Giancarlo Mariani

UN FILE SEQUENZIALE DIVENTA RELATIVO!

Teoria e, soprattutto pratica, per memorizzare dati su disco mediante gestione di files pseudo - relativi

```
/* CREA.C - Borland's Turbo-C 2.0 */
/* Crea il file "nomi.txt" e permette di inserire una li-
sta di nomi */
```

Include: Questi files verranno "inclusi" nel testo sorgente C; conterranno tutte le dichiarazioni necessarie al buon funzionamento delle istruzioni C.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
```

Main program: i termini **void main()** indicano l'inizio del programma principale; void, in particolare, specifica che il main non avrà parametri di ritorno.

```
void main()
{
```

Dichiarazione variabili: le variabili **stream**, **i** e **Stringa** sono dichiarate all'interno del main, quindi visibili solo da questo.

```
   FILE *stream;
   int i;
   char Stringa[81];
```

clrscr( ) cancella lo schermo.

```
   clrscr();
   printf("Nome e cognome separati da spazio (* end)\n");
```

L'istruzione **fopen** serve per aprire un file. I parametri sono il **nome** ed il **modo**, che in questo caso (**wt**) specifica che il file è di testo (t) e che è aperto in scrittura (w). Altri modi sono: **rt** = file di testo in lettura; **r+t** = file di testo in r/w, ed altri. Nella variabile **stream** verrà trascritto il puntatore al file, che permetterà di utilizzare le istruzioni C per lavorare su file stesso. Se il C non riesce ad aprire o a creare il file, in stream metterà il valore **NULL**.

Il listato "Crea.C" (prima parte)
I commenti posti nei riquardi saranno di aiuto per comprendere la tecnica adoperata per realizzare il file

Nelle precedenti chiacchierate sul C abbiamo avuto modo di esaminare il trattamento di **files sequenziali** (cfr. articoli su ordinamento, microcad, ecc.), con semplici programmi che consentivano di creare, leggere e scrivere, appunto, files sequenziali.

In queste pagine approfondiremo la conoscenza di tali files, cominciando a parlare di files relativi. In realtà i files usati nei programmi proposti non saranno utilizzati proprio come se fossero relativi, ma ancora come sequenziali, ma dotati della caratteristica di offrire la possibilità di posizionarsi all'interno del file stesso in un qualsiasi punto voluto.

Come programma di riferimento è stato preso **BubbleSort**, pubblicato in Gw Basic sul **N. 77** di C.C.C. (a proposito sugli ordinamenti di vettori), opportunamente implementato in C e modificato in modo tale da non ordinare un array posto in memoria, ma un file di testo memorizzato direttamente su disco.

Il primo listato

Il primo programma da esaminare si chiama **CREA.C**, consente di inserire una lista di nomi e di salvarla nel file **Nomi.txt**. Il listato è molto simile a quello presentato nell'articolo di C.C.C. citato, ma possiede una caratteristica particolare.

Tutti i **record**, ossia i nominativi inseriti, prima della registrazione vengono portati alla lunghezza di **40** caratteri. Ciò significa che se l'operatore inserisce nominativi di lunghezza minore, il programma auto-

maticamente aggiungerà spazi per raggiungere i 40 caratteri, anzi, 41 (ad esempio, 20 per il cognome, uno spazio, 20 per il nome).

Cominciamo a digitare il programmino *Crea* ed a farlo girare. Se non sono stati commessi errori, questo chiederà di inserire un nominativo. A questo punto dovremo inserire il **Cognome**, uno spazio di separazione ed infine il **Nome**, tutto questo per una lunghezza massima di 41 caratteri.

Crea provvederà a registrare, sul file *Nomi.txt*, tutti i nominativi inseriti. Per terminare l'inserimento sarà sufficiente digitare un asterisco (*) e premere il tasto Return.

A questo punto si può scrivere il programma **ORDINAF.C** e farlo girare.

Il programma esegue le seguenti operazioni:

```
   stream = fopen("nomi.txt","wt");
```

Le righe **if strlen**... e **while strlen**... aggiungono spazi in fondo alla stringa inserita, fino a raggiungere la lunghezza di 41 caratteri.

```
   if (stream != NULL)
   {
     i=1;
     do
     {
       printf("Numero %d : ",i);
```

gets: permette di inserire una stringa da tastiera, e la inserisce nella variabile stringa specificata.

```
       gets(Stringa);
      if (strlen(Stringa)<41)
       while (strlen(Stringa)<41) strcat(Stringa," ");
```

fprintf scrive nel file la stringa inserita.

```
       if (Stringa[0]!='*') fprintf(stream,"%s",Stringa);
       i++;
     }
```

Il ciclo **while**... permette di ripetere il tutto sino all'inserimento di un asterisco.

```
     while (Stringa[0]!='*');
   }
```

fclose chiude il file specificato.

```
   fclose(stream);
}
```

Il listato "Crea.C" (continuazione e fine)

I commenti posti nei riquardi saranno di aiuto per comprendere la tecnica adoperata per realizzare il file

1) Conta i nominativi presenti nel file Nomi.txt.

2) Visualizza, dopo conferma, i nominativi.

3) Ordina alfabeticamente i nominativi DIRETTAMENTE SUL FILE, senza caricarli in memoria.

4) Visualizza, dopo conferma, i nominativi ordinati.

Al termine del programma avremo il file Nomi.txt ordinato alfabeticamente. E' importante che tutti i record, ossia le righe, del file siano lunghe 41 caratteri, pena l'errato ordinamento del file.

Bubble Sort

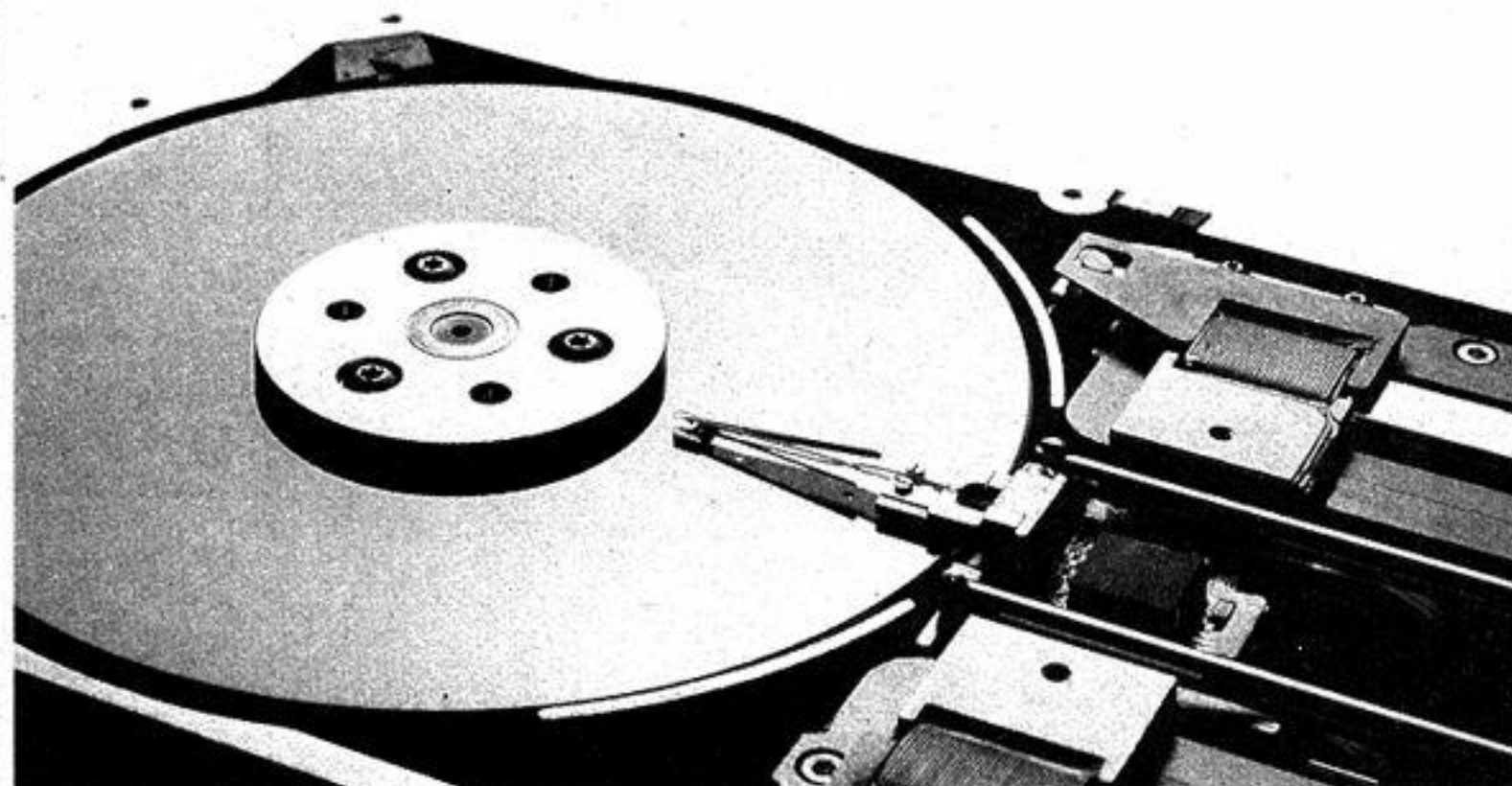
Il programma **ORDINAF** è costruito in modo da ordinare alfabeticamente un file di testo con record lunghi **41** caratteri, creati, appunto, da *Crea*. Il programma ordina il file secondo il metodo **Bubble Sort**; per chi non ricordasse come lavora questo tipo di ordinamento, ecco una rinfrescata:

Bubble Sort è il metodo più conosciuto per ordinare una lista di nomi o numeri. La sua popolarità deriva dal fatto che l'algoritmo di ordinamento è estremamente semplice e si basa soltanto su due cicli di tipo For. Il metodo si basa su ripetuti confronti ed eventuali scambi di elementi adiacenti. Al primo ciclo l'elemento più grande viene gradatamente "portato" all'ultima posizione dell'array; al

secondo ciclo il secondo elemento più grande viene portato alla penultima posizione, e così via fino a raggiungere la lista ordinata.

Dal momento che l'algoritmo di Bubble Sort è guidato da due cicli For, il numero di confronti (istruzione **IF**), eseguiti per ordinare una lista, non dipenderà dallo stato iniziale della lista stessa (ossia, se era completamente disordinata o già parzialmente ordinata), essendo in ogni caso uguale a...

0.5 * (n * n - n)

...in cui "**n**" è il numero di elementi da ordinare. Il tempo impiegato per ordinare la lista è proporzionale al prodotto "**n * n**". Si può capire, quindi, che Bubble Sort sarà molto lento quando gli elementi da

```
/* ORDINAF.C - Borland's Turbo-c 2.0 */
/* Ordina "nomi.txt", direttamente su disco */
```

Include: sono i files che verranno "inclusi" nel testo sorgente C e che contengono tutte le dichiarazioni necessarie al buon funzionamento delle istruzioni C.

```
#include <stdio.h>
#include <conio.h>
#include <string.h>
```

Variabili globali: sono le variabili che verranno "viste" dall'intero programma C, ossia dal Main e dalle procedure e funzioni presenti. In questo caso, la sola variabile globale è "stream", che è dichiarata di tipo FILE.

```
/* Variabili Globali */

FILE *stream;
```

Prototipi di procedure e funzioni: queste righe definiscono quante e quali procedure e funzioni saranno usate nel programma, nonchè il numero ed il tipo di parametri ad esse connessi. Questo particolare permette al compilatore di eseguire un controllo RUN- TIME (durante l'esecuzione) dei parametri passati alle procedure e funzioni, elaborazione altrimenti non possibile.

```
void VediNomi(void);
void BubbleSort(int c);

/* Routine ordinam. "nomi.txt" con BubbleSort */
void BubbleSort(int c)
{
```

Dichiarazione variabili locali, che quindi verranno viste solo dalla procedura in questione.

```
   int a,b,Pos1,Pos2;
   char x[45],y[45];
```

"fopen", come visto prima, apre il file. Il parametro "r+t" indica che il file deve essere aperto in lettura e scrittura.

```
  stream = fopen("nomi.txt","r+t");
```

La routine di ordinamento "BubbleSort" è la procedura che ordina il file, in cui vengono dichiarate alcune variabili locali, che non verranno viste quindi dal resto del programma.

La tecnica per l'ordinamento è la seguente: sappiamo che il file **Nomi.txt** è composto da tanti (finti) record lunghi **41** caratteri ciascuno.

Per recuperare il record desiderato, quindi, ci si dovrà posizionare nel file ad una posizione data da **41 * Numero record**, riferita all'inizio del file.

Ad esempio, il primo record (il numero 0) si troverà a partire dalla posizione **0**, il secondo dalla **41**, il terzo dalla **82**, e così via.

Per ordinare una coppia di record, secondo il metodo Bubble Sort, bisognerà procedere nel modo seguente:

- Caricare due record adiacenti
- Eseguire il confronto tra i due record caricati
- Se il primo è maggiore del secondo, eseguire lo scambio tra i due record, ossia, registrare il primo nella posizione del secondo e viceversa.

Per compiere tale operazione avremo bisogno di una istruzione che consenta di spostarci a piacimento nel file.

Si tratta dell'istruzione **fseek**, la cui sintassi è riportata nell'apposito commento posto nel programma a lato.

Il listato di ordinamento (prima parte)

I cicli "for a" e "for b" sono i due cicli sui quali si basa la routine di bubble sort. Il primo è un ciclo che parte dal secondo elemento fino all'ultimo - 1; il secondo è il ciclo che parte dall'ultimo elemento fino all'elemento indicato dal primo ciclo.

```
for (a=2; a<=c-1; a++)
 {
   for (b=c; b>=a; b--)
   {
```

Pos1 e Pos2 calcolano gli spostamenti per il record indicato dal ciclo più interno (for b...) e per quello precedente. La moltiplicazione per 41 tiene conto, come detto prima, che ogni record è lungo 41 caratteri.

```
    Pos1=41*(b-2); Pos2=41*(b-1);
```

La sintassi di **fseek** è la seguente:
fseek (FILE, Posizione, DaDove)
in cui **FILE** è il file al quale ci si riferisce (nel nostro caso, "stream").
Posizione indica la posizione nella quale spostarci; nel nostro caso potrà valere 0, 41, 82, 123, eccetera.
DaDove rappresenta l'indice a partire dal quale si desidera effettuare lo spostamento. In questo caso, **SEEK_SET** indica che lo spostamento è riferito all'inizio del file. Assegnando, ad esempio, alla variabile **Posizione** il valore 82, fread sposterà, il puntatore del file, 82 bytes in avanti rispetto all'inizio. Cambiando il parametro **DaDove** è possibile eseguire spostamenti relativi alla fine del file o alla posizione corrente.

```
   fseek(stream,Pos1,SEEK_SET); fgets(x,42,stream);
```

"fseek...fgets" posizionano il puntatore del file nelle posizioni calcolate prima e caricano i due record nelle variabili stringa "x" e "y".

```
   fseek(stream,Pos2,SEEK_SET); fgets(y,42,stream);
```

"stricmp" serve per confrontare le due stringhe. Se sono uguali, stricmp darà come risultato 0; se la prima è maggiore della seconda, il risultato sarà maggiore di zero, mentre risulterà negativo se la prima è minore della seconda.
Se la prima stringa è maggiore della seconda significa che gli elementi x ed y sono posti in ordine errato, quindi da scambiare tra loro. Ciò si ottiene semplicemente registrando, nel file, x al posto di y e viceversa. L'operazione è ottenuta dalle istruzioni "fseek....fprintf", poste all'interno di "if".

```
     if (stricmp(x,y)>0)
     {
       /* Scambio x con y */
       fseek(stream,Pos1,SEEK_SET); fprintf(stream,"%s",y);
       fseek(stream,Pos2,SEEK_SET); fprintf(stream,"%s",x);
```

La serie di parentesi graffe chiuse "}" chiude l'if ed i vari cicli for.

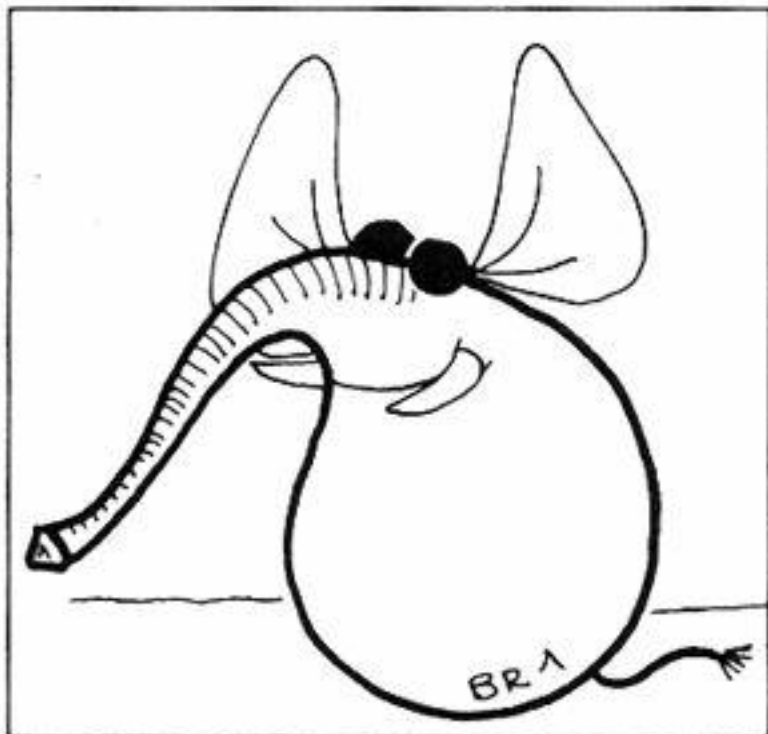
```
     }
   }
 }
```

"fclose" chiude il file specificato.

```
  fclose(stream);
}

/* visualizza i nomi */
```

VediNomi. Questa routine serve soltanto a mostrare il contenuto del file "nomi.txt" sul video.
L'istruzione "while fgets...." continua a leggere i record dal file, sino a quando viene incontrata la fine del file stesso, ossia, fino a quando l'istruzione fgets restituisce il valore NULL.

```
void VediNomi()
{
 char Stringa[45];
```

Il listato di ordinamento (seconda parte)

ordinare risulteranno esser numerosi. Nulla vieta al lettore, però, di implementare un metodo più efficiente, magari tra quelli descritti nello stesso numero 77 di C.C.C.

Ripetiamo che la principale differenza tra il bubble sort presentato in queste pagine e quello del n. 77, risiede nel fatto che questo non ordina un array presente nella **memoria Ram**, ma effettua un ordinamento direttamente sul contenuto di un file.

Ai lettori, come al solito, il compito di sofisticare il programma per renderlo più versatile ed adatto alle proprie esigenze. L'idea più ovvia è certamente quella di implementare altri tipi di ordinamento alfabetico, più veloci ed efficienti.

Il limite, nel caso specifico, è rappresentato dall'hardware (cioè: la coppia controller / drive) che, a dispetto di procedure s/w rapidissime, potrebbe non evidenziare particolari vantaggi offerti da un programma particolarmente efficace.

La lentezza di risposta delle parti meccaniche di un drive, infatti, rende spesso vani gli sforzi dei programmatori.

```
  printf("\n");
  stream = fopen("nomi.txt","rt");
  while (fgets(Stringa,42,stream) != NULL) printf("%s\n",Stringa);
  fclose (stream);
  printf("\n");
}
```

Main program: i termini "void main()" indicano l'inizio del main program e void specifica che il main non avrà parametri di ritorno.

```
void main()
{
```

Dichiarazione variabili locali, viste solo nel main.

```
   int c,k;
   char Stringa[45];

   clrscr();
   printf("Un momento, sto contando i nomi\n");
```

Parte di conteggio nomi: funziona come la visualizzazione dei nomi, solo che invece di scrivere la stringa caricata da disco, incrementa un contatore.

```
   stream = fopen("nomi.txt","rt");
   c=0;
   if (stream != NULL)
   {
      while (fgets(Stringa,42,stream) != NULL) c++;
      fclose (stream);
   }
   printf("Il file 'nomi.txt' contiene %d nomi\n",c);
   printf("Li vuoi vedere (s/n) : ");
```

L'istruzione **scanf** permette di inserire una variabile da tastiera. Il parametro **%s** indica che la variabile da inserire è di tipo stringa.

```
   scanf("%s",Stringa);
   if (Stringa[0]=='s' || Stringa[0]=='S') VediNomi();
   printf("Un momento, sto ordinando i nomi\n");
```

Le istruzioni **VediNomi** e **BubbleSort** servono, appunto, per richiamare le procedure omonime.

```
   BubbleSort(c);
   printf("Vuoi vedere i nomi ordinati (s/n) : ");
  scanf("%s",Stringa);
   if (Stringa[0]=='s' || Stringa[0]=='S') VediNomi();
}
```

Il listato di ordinamento (continuazione e fine)

LE ISTRUZIONI JMP, CALL, RET

by Giancarlo Mariani

Tre istruzioni dell'Assembler del PC sono utili sia per modificare le normali esecuzioni del programma, sia per realizzare funzioni ripetitive

Istruzione JMP

JMP è il codice mnemonico (Assembler) dell'istruzione, abbreviazione della parola inglese **"jump"**, ossia salta. Lo scopo di JMP è, infatti, quello di saltare alla parte di programma specificata. Il salto è di tipo **incondizionato**, ossia viene effettuato in qualsiasi caso, senza tener conto di alcun flag o parametro.

Destinazione. E' l'indirizzo al quale il programma deve saltare. Destinazione può essere specificato da:

- Un indirizzo scritto in modo **assoluto** (quasi mai usato nei compilatori Assembler).
- Un'**etichetta** corrispondente al punto di programma al quale saltare (forma sintattica più comune).
- Il registro **CX**, che sostituisce l'indirizzo assoluto al quale saltare.
- Il registro **BX** che funge da puntatore ad una **tabella** specificata, dalla quale si potranno leggere gli indirizzi ai quali saltare.

Tutti gli indirizzi specificati nell'istruzione JMP fanno ovviamente riferimento al **code segment CS**.

SHORT. E' un parametro *opzionale* che indica, se usato, che il salto effettuato da JMP è compreso tra **-128** e **+128** bytes dal punto chiamante (salto **relativo**).

Questa forma permette di risparmiare un byte nell'istruzione e di velocizzare l'esecuzione della stessa.

Ovviamente, in caso di salti più lunghi di +/- 128 bytes, non sarà possibile usare tale parametro.

Le istruzioni di cui ci occuperemo in queste pagine sono:

JMP, che salta ad una parte di programma specificata;

CALL che richiama una subroutine;

RET, posta alla fine della subroutine richiamata, che consente il ritorno al programma chiamante.

Alcuni esempi di istruzione JMP:

```
JMP   5500h              ; Indirizzo assoluto
JMP   Parte2             ; Etichetta
JMP   SHORT Vicino       ; Salto relativo.
JMP   CX                 ; Indirizzo in CX
MOV   [BX].Tabella       ; Salto tab. punt. BX
```

JMP; attenzione a:

La parte di programma alla quale si salta deve assolutamente contenere istruzioni Assembler 8086; se, infatti, per un errore di calcolo il salto càpita in una parte di memoria non contenente un programma (ma, ad esempio, un dato), il computer si può inchiodare. L'istruzione JMP non influenza alcun Flag.

Istruzione CALL

CALL è il codice mnemonico dell'istruzione, ed in inglese significa **chiama**. Lo scopo di CALL è quello di attivare la subroutine specificata. La chiamata è incondizionata, ossia viene effettuata in qualsiasi caso, senza tener conto di alcun flag o parametro.

Alcuni esempi di istruzione CALL:

```
CALL 5500h            ; Indirizzo assoluto
CALL Parte2           ; Etichetta
CALL AX               ; Indirizzo in AX
CALL Tabella.[SI]     ; Salto tabellare puntato da SI
```

Destinazione. E' l'indirizzo corrispondente all'inizio della subroutine che il programma deve chiamare. Destinazione può essere specificato da:

- Un indirizzo scritto in modo **assoluto** (quasi mai usato nei compilatori Assembler).
- Un'**etichetta** corrispondente all'inizio della procedura da chiamare (forma più comune).
- Il registro **AX**, che sostituisce l'indirizzo assoluto al quale saltare.
- Il registro **SI** che fa da puntatore ad una tabella specificata, dalla quale si potranno leggere gli indirizzi di inizio delle varie procedure da chiamare.

Tutti gli indirizzi specificati nell'istruzione CALL fanno ovviamente riferimento al code segment **CS**.

Attenzione a:

Anche la procedura richiamata con CALL deve contenere corrette istruzioni Assembler 8086; in caso contrario il computer si può inchiodare.

L'etichetta specificata può riferirsi ad una procedura appartenente allo stesso segmento nel quale è contenuta l'istruzione (**NEAR_PROC**), oppure ad una procedura scritta "fuori" dal segmento (**FAR_PROC**). In quest'ultimo caso l'istruzione occuperà 2 bytes in più e sarà più lenta nell'esecuzione.

Al momento della chiamata, nello **Stack** viene memorizzato l'indirizzo di ritorno, che sarà quello corrispondente all'istruzione successiva a CALL, e che verrà utilizzato da RET per ritornare al programma chiamante nel punto corretto. L'indirizzo è composto da 2 bytes in caso di chiamata ad una subroutine posta nello stesso segmento; in caso di subroutine in altro segmento i bytes saranno 4.

L'istruzione CALL non influenza alcun Flag.

L'istruzione RET

RET è il codice mnemonico dell'istruzione, ed è l'abbreviazione della parola inglese **return**, dall'ovvio significato. Lo scopo di RET, posto alla fine di una subroutine, è quello di consentire il ritorno al punto di programma successivo alla chiamata della subroutine stessa.

Il ritorno è incondizionato, ossia viene effettuato in qualsiasi caso senza tener conto di alcun flag o parametro.

[pop_value]. Parametro opzionale che indica il numero di bytes di cui è composto l'indirizzo di ritorno da prelevare dallo **Stack**. E' un'opzione particolare, abbastanza difficile da capire ed usare; per il momento se ne evita la spiegazione, rimandandola ai prossimi fascicoli.

Attenzione a:

RET deve, ovviamente, essere l'**ultima** istruzione presente nella procedura chiamata. Eventuali altre istruzioni poste dopo RET non verranno eseguite. L'indirizzo dal quale riprendere l'esecuzione viene prelevato dallo Stack; quest'ultimo, pertanto, non deve essere manomesso prima dell'istruzione. Nel caso in cui la subroutine sia stata chiamata dallo stesso segmento nel quale è presente la subroutine stessa, l'indirizzo di ritorno sarà composto da 2 bytes e RET preleverà 2 bytes dallo stack. Nel caso in cui la subroutine sia stata chiamata da un segmento diverso, l'indirizzo sarà composto da **4** bytes, che verranno ancòra prelevati dallo stack. Il parametro opzionale permette di stabilire il numero di bytes da prelevare dallo stack, allo scopo di non usare il default dell'istruzione. Ad esempio si possono prelevare 4 bytes in caso di ritorno nello stesso segmento; oppure 2 bytes in caso di ritorno ad un altro segmento. Un'operazione del genere è molto pericolosa e, se non usata correttamente, può portare all'inchiodamento del computer. Risulta utile, del resto, in alcuni particolarissimi casi. L'istruzione RET non influenza alcun flag.

Due esempi di istruzione RET:

```
RET    ; Ritorna
RET 4; Ritorna prelev. un indir.
```

Dissertazioni

L'istruzione **CALL**, prima di effettuare il salto, pone l'indirizzo di ritorno nello Stack; questo sarà prelevato da **RET** in modo che il programma possa ritornare esattamente nel punto in cui era stato interrotto da Call.

Per questo motivo le subroutines **NON DEVONO ASSOLUTAMENTE** alterare il contenuto dello stack, dello stack pointer (**SP**) e dello stack segment (**SS**). In caso contrario bisogna aver cura di rimettere i valori originari **prima** di impartire l'istruzione Ret, che altrimenti preleverebbe un indirizzo di ritorno errato, con conseguenze facilmente immaginabili.

Pericolose manipolazioni dello stack possono venire involontariamente generate tramite le istruzioni **Push**, **Pop** (che vedremo prossimamemte), **Mov SS, xxxx** (che modifica lo Stack Segment); oppure **Mov SP, xxxx**, che modifica lo Stack Pointer.

Per il momento, evitate l'uso di tali istruzioni nelle vostre subroutine. Più avanti verrà spiegato come sfruttare lo stack non solo per contenere gli indirizzi di ritorno, ma anche per altre utili funzioni.

Un programma con JMP

L'esempio per l'istruzione Jmp è semplicissimo, dal momento che non necessita di parametri particolari. Riferendoci all'intestazione per i programmi **COM**, vista nel numero scorso, aggiungiamo alcune istruzioni in modo che il programma diventi come indicato nel riquadro qui a lato.

```
;
; SALTO.ASM : Esempio di istruzione JMP
; Somma 1230h con 4526h e mette il risultato in RESULT.
;
cseg        SEGMENT PARA PUBLIC 'CODE'
            org 100h
            ASSUME cs:cseg, ds:dseg, ss:cseg, es:cseg

Start:
            mov   AX,CS
            mov   DS,AX              ;DS = CS
            mov   ES,AX              ;ES = CS

; Programma di somma
            mov   AX,1230h           ;AX=1230h
            mov   BX,4526h           ;BX=4526h

            jmp   Somma              ;Salta a label "Somma"

            mov   AX,5733h           ;Queste istruzioni NON
            mov   BX,3412h           ;verranno MAI eseguite.

Somma:      add   AX,BX              ;Esegue AX=AX+BX
            mov   RESULT,AX          ;Risultato in RESULT

            mov   AH,4Ch
            mov   AL,0
            int   21h                ;Ritorno al DOS

; Dati utilizzati dal programma
            RESULT    DW 0           ;Riserva 2 bytes

cseg        ENDS
            END   Start
```

Le due istruzioni **mov AX,5733h** e **mov BX,3412h** non verranno mai eseguite durante l'esecuzione del programma; infatti risultano escluse a causa della presenza dell'istruzione Jmp.

Si tralascia la descrizione del programma, grazie alla sua semplicità; per le rimanenti istruzioni si faccia riferimento al fascicolo precedente.

Alcune note merita, a parte, la **direttiva DW**. Questa **NON** è un'istruzione Assembler, ma è una direttiva che comunica al compilatore di riservare due bytes di memoria; ricorderete certamente la **DB** pubblicata sul n. 81, che però riservava un solo byte. Dopo aver digitato il programmino, lo si può registrare con il

nome **SALTO.ASM** e quindi lo si può compilare tramite le istruzioni:

MASM SALTO;
LINK SALTO;
EXE2BIN SALTO.EXE SALTO.COM
DEL SALTO.EXE

Mandatelo in esecuzione con il comando **Salto** pur se, ovviamente, non potremo osservare alcun risultato sul video. Per verificare l'effettivo comportamento del programma dovremo ancora una volta servirci del **CodeView Debugger CV**.

Prima di richiamare il debugger, dovete ricompilare il programma usando le opzioni del MASM e del LINK che forniscono, al debugger, le informazioni necessarie per visualizzare il listato sorgente, comprese tutte le variabili ed i commenti relativi.

MASM /Zi SALTO;
LINK /Co SALTO;

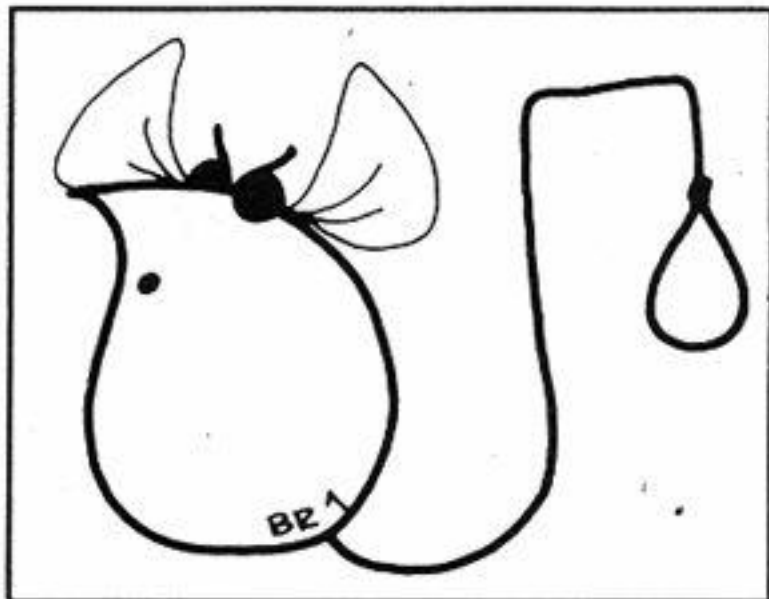
Questa volta non trasformiamo il file **.EXE** in **.COM**, perchè altrimenti perderemmo le informazioni sul sorgente da passare al debugger; richiamate direttamente CV con il comando:

CV SALTO.EXE

Fate ora partire il programma Salto, premendo i tasti **Alt + R** che produrranno la visualizzazione di un menu a tendina; scegliete l'opzione **Restart** per posizionare il registro **IP** (Instruction **P**ointer) all'inizio del programma (che ricordiamo, è messo a 0100h dalla direttiva ORG). Ora, premendo il tasto **F10**, potrete eseguire il programma "passo-passo", ossia un'istruzione alla volta, per osservare i risultati contenuti nei vari registri (visualizzati sulla destra del programma). In particolare osserverete che, al raggiungimento dell'istruzione Jmp, il registro IP viene caricato con un nuovo valore, ed il programma salta all'etichetta specificata.

```
;
; SOMMASUB.ASM : Somma di due numeri tramite subroutine
; Primo numero   = locazione OP1
; Secondo numero = locazione OP2
; Risultato      = locazione RESULT
;
cseg        SEGMENT PARA PUBLIC 'CODE'
            org 100h
            ASSUME cs:cseg, ds:dseg, ss:cseg, es:cseg

Start:
            mov  AX,CS
            mov  DS,AX              ;DS = CS
            mov  ES,AX              ;ES = CS

            mov  AL,OP1             ;Primo operando
            mov  AH,OP2             ;Secondo operando

            call Somma              ;Esegue la somma

            mov  AH,4Ch
            mov  AL,0
            int  21h                ;Ritorno al DOS

; Procedura "Somma": eseguela somma dei duenumeri ; con-
tenuti nei registri AH e AL, e mette il risultato
; nella locazione RESULT

Somma:
            add  AL,AH             ;Addiziona AL con AH
            mov  RESULT,AL         ;Salva AL in RESULT
            ret                    ;Ritorna al prog. chiamante

; Dati utilizzati dal programma
            OP1       DB 10
            OP2       DB 15
            RESULT    DB 0

cseg        ENDS
            END  Start
```

Un listato con Call e Ret

Il programma esaminato nel numero scorso (somma di due numeri) viene presentato, nel riquadro a lato, in modo tale che la somma sia eseguita da una subroutine. Tralasciando la spiegazione sulle intestazioni e sul caricamento degli operandi nei registri (fasi già descritte nel

fascicolo precedente) ci limiteremo a dire che, una volta caricati i valori nei registri **AL** e **AH**, viene richiamata la procedura **Somma**, che somma i due valori e pone il risultato in **Result**. La procedura viene richiamata con l'istruzione **Call** e termina con **Ret**, che effettua il ritorno al programma principale, esattamente in corrispondenza dell'istruzione successiva a Call.

Si sottolinea che la procedura *Somma* può essere usata in infinite parti del programma, tenendo conto che i valori da sommare devono essere posti in AL e AH, ed il risultato lo ritroveremo in Result. Questo, ovviamente, evita di scrivere le istruzioni ogni volta che occorre, sostituendole con una semplice Call.

Per una procedura semplice come quella presentata nel riquadro, il risparmio di memoria non è molto evidente. Quando, però, la procedura diventa consistente, si nota con evidenza la convenienza di Call... Ret.

Ovviamente bisogna fare in modo che le procedure scritte siano facilmente richiamabili da più punti del programma, senza esser costretti ad adattarle ad ogni occasione.

Si consiglia di assegnare parametri in ingresso facilmente accessibili (in questo caso, i registri AL ed AH) come quelli di uscita (in questo caso, Result), di modo che qualsiasi punto di programma che richiami la subroutine sappia il punto in cui inserire i parametri o da cui prelevare i risultati.

Digitato il programmino, registriamolo con il nome **SOMMASUB.ASM** e compiliamolo tramite le istruzioni:

MASM SOMMASUB;
LINK SOMMASUB;
EXE2BIN SOMMASUB.EXE SOMMASUB.COM
DEL SOMMASUB.EXE

....e mandiamolo in esecuzione tramite il comando:

SOMMASUB

Anche in questo caso non potremo osservare i risultati sul video, ma dovremo servirci di CV. Nonostante l'uso di CV sia già stato spiegato precedentemente, rinfrescheremo la memoria presentando un riassunto su alcune istruzioni del Debugger.

Al richiamo, il programma presenterà una schermata in cui sono visualizzati, al lato destro (posti in verticale) tutti i **registri** dell'8086 e relativi contenuti in esadecimale; in basso una finestra per i **comandi immediati**; la parte rimanente dello schermo sarà occupata dal listato **sorgente**.

Il tasto **F6** sposta dalla finestra immediata a quella del sorgente, e viceversa. In quest'ultima finestra, con i soliti tasti cursore, PgUp, PgDn, ecc. potremo **spostarci** a piacere lungo il listato sorgente, mentre nella prima potremo impartire comandi a CV.

Il comando **D start end** produce la visualizzazione del contenuto di una parte di memoria, dall'indirizzo Start all'indirizzo End, espressa in esadecimale. Gli indirizzi "start" e "end" possono essere dati sia come valore assoluto che come simbolico (etichette).

Il comando **E addr.** permette di cambiare il valore della locazione specificata; "addr" può essere un indirizzo dato in modo assoluto oppure simbolico.

Il tasto **F10** produrrà un'esecuzione del programma passo-passo, permettendo di osservare i risultati ed il cambiamento dei vari registri sul video dopo ogni istruzione.

I tasti **Alt-R** e quindi **Start**, produrranno l'esecuzione completa del programma, in modo normale.

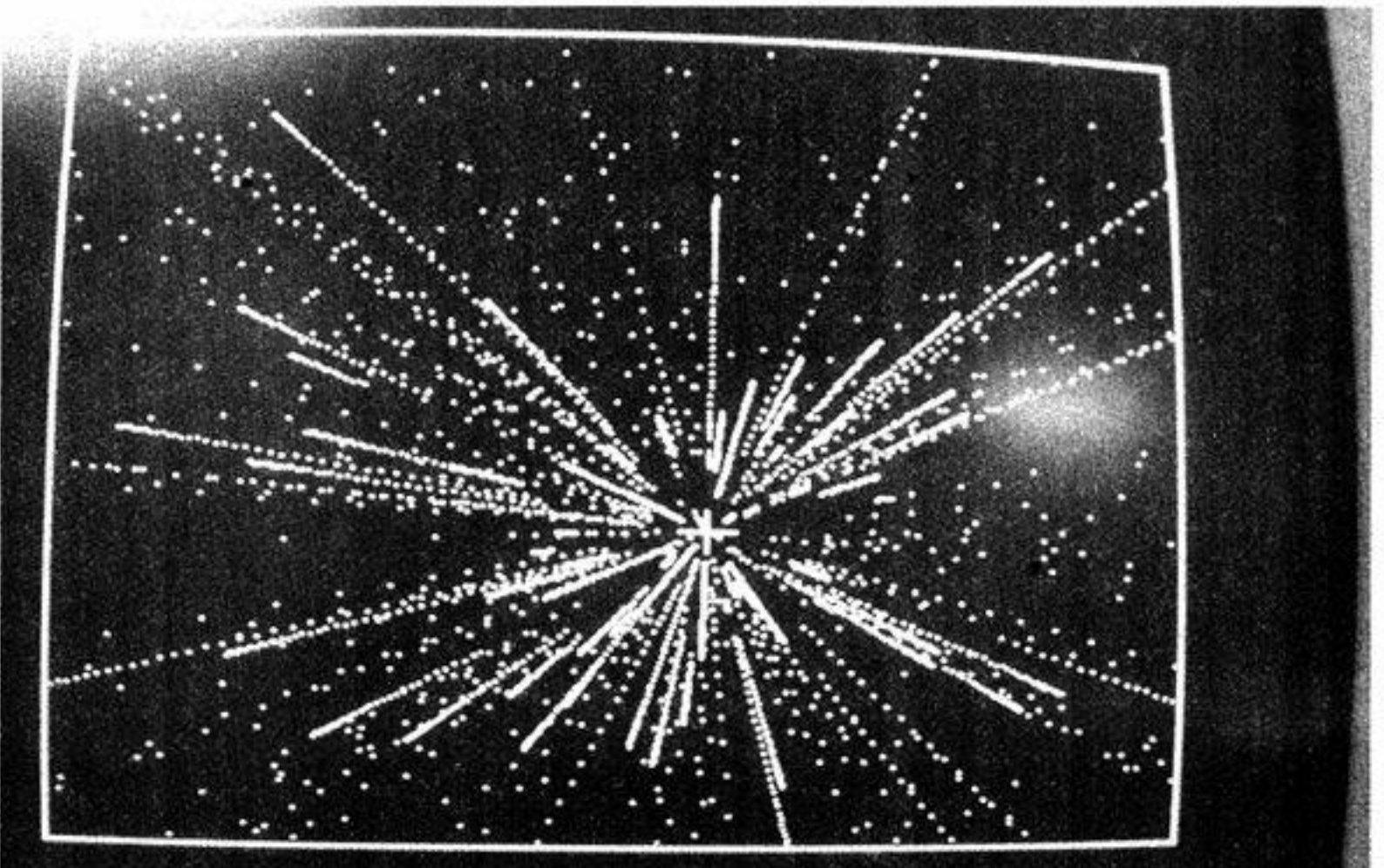
Il comando **Help**, impartito in qualsiasi momento, visualizzerà l'help del debugger, che comprende spiegazioni dettagliate (in inglese) per tutti i comandi e le opzioni presenti in CV.

Infine, il comando **Quit** farà tornare in ambiente **Dos**.

Ricordiamo che per "debuggare" un programma con **CV**, questo deve essere prima compilato e linkato con opzioni particolari, che producono una parte di simboli che serviranno a CV per controllare direttamente il listato sorgente. Omettendo tali opzioni si potrà comunque debuggare il programma, ma in modo assoluto, ossia con gli indirizzi al posto delle etichette, e senza commenti.

Concludiamo la chiacchierata ricordando che le opzioni che permettono di compilare un programma per CV, sono:

MASM /Zi NOMEPROG;
LINK /Co NOMEPROG;

Si richiama CV, infine, tramite
CV NOMEPROG.EXE

di Alessandro de Simone

L'ALTRO MODO DI DIRE PASCAL

Nel mondo Ms - Dos il linguaggio Pascal è uno dei più noti ed usati. La prestigiosa Microsoft propone, ovviamente in italiano, una sua versione del popolare compilatore.

Quali sono le caratteristiche che spinge un potenziale utente a preferire un programma piuttosto che un altro?

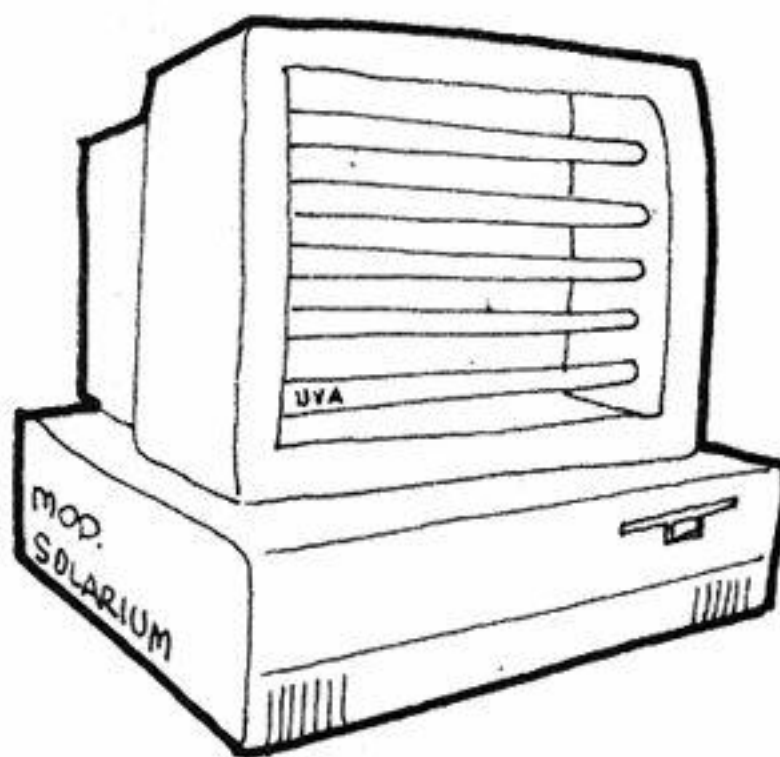
Senza ombra di dubbio possiamo affermare che le risposte sono poche, ma essenziali; anzitutto c'è il prestigio goduto dalla software house che propone il pacchetto. Tra i word processor di pubblico dominio, ad esempio, vi sono numerosi programmi che funzionano benissimo, ma sono pochi gli utenti che li utilizzano intensivamente, magari in modo professionale. Chi può garantire, tra l'altro, eventuali **upgrade** quando, scoperti gli inevitabili limiti del programma, si desidera una versione più aggiornata e potente? Soltanto s/w house rinomate sono in grado di offrire un minimo di assistenza ai propri clienti ed è questo il motivo principale per cui molti programmi, benchè apparentemente validi, non possono reggere il confronto se paragonati a prodotti più blasonati.

Per quanto riguarda (e torniamo al discorso iniziale) gli altri motivi che decretano il successo di un prodotto, tra questi figurano il rispetto di eventuali standard richiesti dall'utenza professionale e (siamo in Italia, ragazzi!) la disponibilità di consultare **manuali in italiano.**

QuickPascal della **Microsoft** offre tutte le caratteristiche accennate, ad un prezzo decisamente contenuto ed alla portata anche di uno studente.

La confezione QuickPascal

La confezione originale Microsoft contiene **tre** dischetti (ci riferiamo al formato 3.5, ma sono disponibili anche i dischetti 5.25) con i quali è possibile installare su disco rigido tutti i files necessari per la programmazione. Inutile dire che anche chi non dispone di hard disk potrà usare QuickPascal; la procedura per realizzare dischetti di lavoro è ovviamente descritta con la massima cura sul manuale.

E su questo (cioè sul manuale accluso alla confezione originale) vogliamo ora intrattenerci. Avete capito benissimo: si tratta di un unico (pur se voluminoso) volume in lingua italiana che descrive con una certa cura tutte le operazioni da compiere per sfruttare al massimo tutte le potenzialità di QuickPascal. In effetti è

anche presente un secondo volumetto (dall'accattivante titolo di "**Up and Running**"), scritto in inglese, che rappresenta un riassunto del manuale più grande. Chi non conosce l'inglese, comunque, non si perde nulla dal momento che tutte le informazioni ivi contenute sono rintracciabili (in forma addirittura... estesa) sull'altro volume.

Il manuale QuickPascal

I capitoli fanno costante riferimento al contenuto dei dischetti acclusi alla confezione originale. I programmi di esempio descritti, infatti, sono spesso parti "estratte" dai sorgenti ed alcuni suggerimenti di modifiche usano, come cavia, proprio quei programmi.

Per questo motivo, nell'introduzione, si precisano le convenzioni tipografiche usate e, dal momento che QuickPascal segue le direttive di uno standard internazionale, una bibliografia viene indicata per approfondire vari argomenti.

Chi non conosce assolutamente nulla di Pascal (ma, inutile dirlo, deve conoscere almeno il concetto fondamentale di linguaggio e, magari, sa già programmare in Basic) troverà, nel primo capitolo, tutte le informazioni che possono introdurlo al compilatore.

Già dal primo capitolo si affronta, in pratica, il linguaggio: nel corso dell'intero volume, infatti, vene seguita la tecnica di apprendimento "immediato", che si realizza attraverso la digitazione di **semplicissimi listati**. Questi, oltre a servire come esempi dimostrativi di particolari forme sintattiche, si prestano benissimo per dimostrare l'efficacia dell'**ambiente integrato**, offerto da tutti i prodotti Microsoft.

Alla prima parte appartengono sette capitoli. Nel **primo** e nel **secondo**, dedicati prevalentemente ai **principianti**, si parla di parole chiave, identificatori, costanti, variabili, operatori, tipi di dati ed altri elementi basilari per una corretta introduzione al linguaggio. Brevissimi

Requisiti richiesti

QuickPascal Microsoft richiede, per un corretto funzionamento, di: un computer IBM (o **compatibile**) su cui sia installata la versione Dos **2.1** o successive; un hard disk ed un floppy disk (**3.5** oppure **5.25**) oppure soltanto due floppy disk drives (ma un hd è ovviamente raccomandato; almeno **448 Kbyte** Ram liberi (benchè l'ideale sia la disponibilità di almeno 521 K).

In altre parole, il linguaggio Microsoft gira (potevate dubitarlo?) su qualunque "carretta", di qualunque marca sia...

esempi, come già detto, completano le varie spiegazioni.

Il **terzo** capitolo affronta i punti cardine che caratterizzano la programmazione strutturata: le procedure e le funzioni sono descritte nel modo più efficace, riferendosi, cioè, a brevi esempi pratici. Il **quarto** capitolo, dopo una chiacchierata sugli operatori relazionali e booleani, si intrattiene sulle strutture iteranti (cicli While, Repeat, For) e su quelle di "salti" (If, Else, Case).

Dal **quinto** capitolo inizia la trattazione di argomenti più "tosti" che, però, una volta compresi, daranno la massima soddisfazione ai programmatori. Ci riferiamo ai tipi di dati definiti dall'utente ed alle funzioni First, Last, Succ, Inc ed a tutte la altre di pari importanza. Nel **sesto** capitolo si affrontano le matrici ed i record, mentre nel **settimo** si parla diffusamente delle famose *unità* (standard o di utente) che ampliano considerevolmente il campo di azione del linguaggio.

Nella seconda parte del volume si affrontano (**cap. 8**) gli argomenti legati all'I/O (Readln, Writeln, Crt eccetera). Il **nono** capitolo spiega l'uso dei files di testo (apertura, chiusura, reindirizzamento); il **decimo** tratta i file binari (tra cui la gestione dei files ad accesso casuale). I puntatori sono l'argomento dell'**11mo** capitolo mentre il **12mo** è destinato agli utenti esperti che vogliono sfruttare fino in fondo la potenza del compilatore: gestione della memoria (allocazione e liberazione), rappresentazione interna dei dati, linkaggio, accesso ai parametri e restituzione di un valore e così via.

Nella terza, ed ultima, parte troviamo gli argomenti più interessanti e "di effetto". Ci riferiamo (**cap. 13**) alla grafica ed alla sua gestione: schede grafiche (a proposito, si possono utilizzare senza problemi le schede CGA, EGA, MCGA, e VGA fino a **256 colori**), colori, coordinate, modalità schermo, animazione, finestre. Nel **capitolo 14** si tratta della gestione dei caratteri e nel **15mo** si affronta, in modo sistematico, la **programmazione strutturata** (costrutti, classi, sottoclassi, metodi, eccetera).

Un'appendice in quattro parti (codici Ascii, direttive del compilatore, unità standard, guida rapida) completano il manuale.

Il linguaggio QPascal

Problematico, come al solito, esprimere un giudizio sulle potenzialità offerte dal linguaggio Microsoft rispetto ai Pascal di altre note s/w house (chissà a chi ci riferiamo...).

Possiamo certamente affermare che l'approccio con l'utente è davvero "amichevole" sia per l'**uso del mouse**, sia per la caratteristica, unica nel suo genere, di

Ambiente integrato

In altri numeri della nostra pubblicazione ci è capitato più volte di parlare di "ambiente integrato".

Con questo termine si intende la possibilità di usare una pluralità di procedure senza esser costretti ad "invocarle" da ambiente Dos o mediante virtuosismi vari.

Se esaminate la directory in cui risiede QuickPascal (o un qualsiasi pacchetto integrato) vi accorgete che questa è formata da numerosissimi files, molti dei quali di tipo .EXE oppure di tipo .COM che, come dovreste sapere, sono programmi eseguibili. Se il pacchetto non fosse integrato, l'utente dovrebbe attivare ciascuno di questi uno alla volta, magari digitando vari parametri come ingresso dopo avere annotato gli stessi offerti (in "uscita") dal programma precedente.

Un pacchetto integrato, invece, compie tutte le operazioni in modo totalmente automatico: l'utente, insomma, crede di avere a che far con un unico programma, pur se complesso, che è in grado di svolgere una quantità incredibile di funzioni.

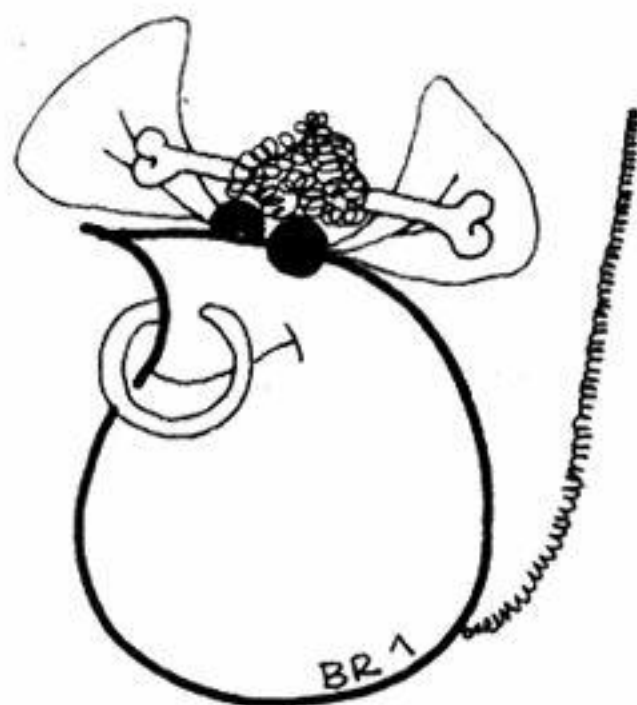
L'integrazione tra le varie potenzialità (digitazione, compilazione, esecuzione, memorizzazione e così via), viene impostata dall'utente premendo un'opportuna sequenza di tasti oppure tramite mouse.

Le procedure seguite, il tipo di menu, l'uso del mouse ed altre caratteristiche che contraddistinguono l'ambiente Microsoft (ben noto agli utilizzatori di altri prodotti Microsoft), sono utilizzate anche in **QuickPascal**. Chi si è già ambientato con il ben noto **QuickBasic**, ad esempio, non avrà alcuna difficoltà a lavorare in QuickPascal dal momento che l'intero "ambiente" risulta straordinariamente simile a quello dell'interprete / compilatore.

visualizzare in diversi colori i vari elementi che caratterizzano un listato sorgente in Pascal. Le parole chiave, infatti, vengono visualizzate in **viola**, i commenti appaiono in **verde**, i nomi delle variabili sono tracciati in **bianco** ed i loro valori in **turchese** (tali colori, inutile dirlo, possono essere modificati a volontà). Si tratta, forse, di un uovo di Colombo, ma vi assicuriamo che avere sullo schermo un colore per ogni "parte" del listato è di una comodità impossibile da descrivere. Soprattutto i principianti (e non soltanto loro) troveranno un giovamento incredibile nel concentrare la propria attenzione, ad esempio, nello "spazio" compreso tra le due parole fatidiche, visualizzate in viola, Begin ed End.

Un'altra caratteristica, di estrema comodità, è rappresentata dalla possibilità di "aprire" più **finestre** in ognuna delle quali si possono caricare programmi sorgenti, o parti di essi. Le dimensioni di ogni finestra, poi, vengono variate usando il mouse; ogni finestra, inoltre, può essere "spentà", occultata, portata in primo piano; il tutto si traduce in un ambiente operativo estremamente **comodo** che, come tale, consente di al programmatore di concentrarsi sul listato da scrivere.

I menu

Otto sono i menu, accessibili da QuickPascal, che gestiscono l'intero ambiente integrato. Il menu **File** permette di caricare (Open), cancellare (New), mescolare (Merge), registrare (Save, Save as), stampare (Print) un programma; oltre a generare una Shell o ad abbandonare definitivamente QuickPascal.

Il menu **Edit** consente principalmente di copiare (Copy), cancellare (Cut), incollare (Paste) parti di programma. **View** controlla la finestra attiva. **Search** effettua ricerche ed eventuali sostituzioni all'interno del testo. **Make**, soprattutto, genera i file eseguibili. **Run** li fa (ri)partire e gestisce eventuali interruzioni (Step, Trace). **Debug**, menu fondamentale, facilita la ricerca di errori di programmazione (gestione dei breakpoints). **Options** consnete di personalizzare, entro certi limiti, l'ambiente di alvoro (velocità mouse, modifica colori schermo). Un menu di aiuto (**Help**) completa l'ambiente. Da sottolineare, comunque, che il menu di Help è sempre disponibile premendo un tasto, soprattutto se è occorso un errore. In ogni caso, posizionandosi sulla parola chiave e premendo **F1**, compare una finestra (gestibile ad **ipertesti**) che illustra brevemente l'errore eventualmente riscontrato (se si è lanciato un programma "difettoso") oppure le potenzialità offerte da una parola chiave (posizionadosi sopra in fase di Edit).

Grazie alle funzioni di taglia e incolla è addirittura possibile "estrarre", da una finestra di esempio, un segmento di programma ed inserirlo nel proprio listato; in seguito, ovviamente, l'utente apporta gli adattamenti del caso.

Conclusioni

Il nome **Microsoft** è, di per se stesso, una garanzia per ciò che riguarda l'affidabilità del prodotto provato.

Inutile dire che un linguaggio **non** può essere giudicato dopo aver fatto girare un paio di programmi, ma richiede un esame più attento e, soprattutto, una comparazione con prodotti analoghi.

Come promesso sul **n. 25**, i listati in Pascal che verranno pubblicati su queste pagine verranno fatti girare anche usando QuickPascal.

Il prezzo del prodotto è di l. **250 mila** (edizione italiana) e di l. **195 mila** (edizione inglese). Il prezzo (IVA esclusa) scende, rispettivamente, a l. **150 mila** e **117 mila** usufruendo dell'offerta speciale per le scuole ed università.

Per maggiori informazioni:
Microsoft S.p.a.
Milano Oltre (palazzo Tiepolo)
Via Cassanese 224
20090 SEGRATE (MI)
Tel. 02/2107.201
Telefax 02/2107.2020

Qui non si possono causare cataclismi naturali, ma si deve progredire nel gioco interagendo a vari livelli, non solo bellici ma anche diplomatici e commerciali, con gli indigeni. In ogni caso, le condizioni metereologiche variano in continuazione, alterando i piani di guerra, gli allevamenti di bestiame e le coltivazioni agricole.

Un capitano sconfitto passa sotto il controllo del giocatore e può guidare il proprio esercito alla conquista di nuove regioni. PowerMonger è suddiviso in **195 isole** con **centinaia** di personaggi dotati di personalità indipendente, la qual cosa garantisce sull'enorma varietà del gioco, veramente unico nel suo genere.

La tecnica

I paesaggi sono proposti secondo poligoni ruotabili, in ogni momento, in tempo reale, pur mantenendo su di essi dei personaggi animati o altri "oggetti" di varia foggia e dimensione. Il sistema di controllo è piuttosto simile, concettualmente, a quello di Populous, con delle icone azionate via mouse.

Gli effetti sonori sono meno monotoni del predecessore, ma comunque molto validi e suggestivi.

Il voto

Un titolo da non perdere, a nessun costo. **10**.

SUBBUTEO

Computer: Amiga inespanso
Gestione: Mouse
Tipo: Simulazione
Softhouse: Electronic Zoo

Il famoso gioco del calcio in punta di dito si è trasformato in un elegante gioco di simulazione, disponibile tra l'altro anche per **C/64**.

Il gioco

Si può disputare una partita contro il computer o contro un amico, oppure svolgere un intero campionato. Si può scegliere la durata della partita, stabilendo i minuti per ogni tempo (sino ai regolamentari **45**), i colori delle magliette dei giocatori ed altro ancora.

L'interfaccia utente è piuttosto efficace: visione **tridimensionale** con punto di vista spostabile a piacimento e possibilità di zoomata. Cliccando su di una propria pedina, e sull'icona del pallone, appare sullo schermo la basetta della pedina, alla "Pool" per intenderci, con un dito controllato dal pouse che si sposta lungo il profilo. Cliccando ancora si sceglie forza ed angolazione del tiro.

Il programma funziona nel pieno rispetto delle regole **FISA** del Subbuteo, come si legge nel manuale oppure da programma stesso, cliccando su di un'apposita icona. Ad esempio, si ha un tempo limitato per effettuare i tiri, non si può tirare in rete oltre una certa distanza, si possono riposizionare i giocatori secondo certe regole fisse eccetera.

La tecnica

La visione tridimensionale del campo di gioco è decisamente gradevole, nitida e precisa. La possibilità di spostamento del punto di vista risulta liberissima ed anche pregevole. Del resto, senza questa precisione tecnica, il gioco non avrebbe certo potuto suscitare molto interesse; ad esempio, anche le animazioni vengono eseguite a 15 fotogrammi al secondo.

Gli effetti sonori sono invece piuttosto trascurati.

Il voto

Una buona simulazione di un gioco non diffusissimo. 8-.

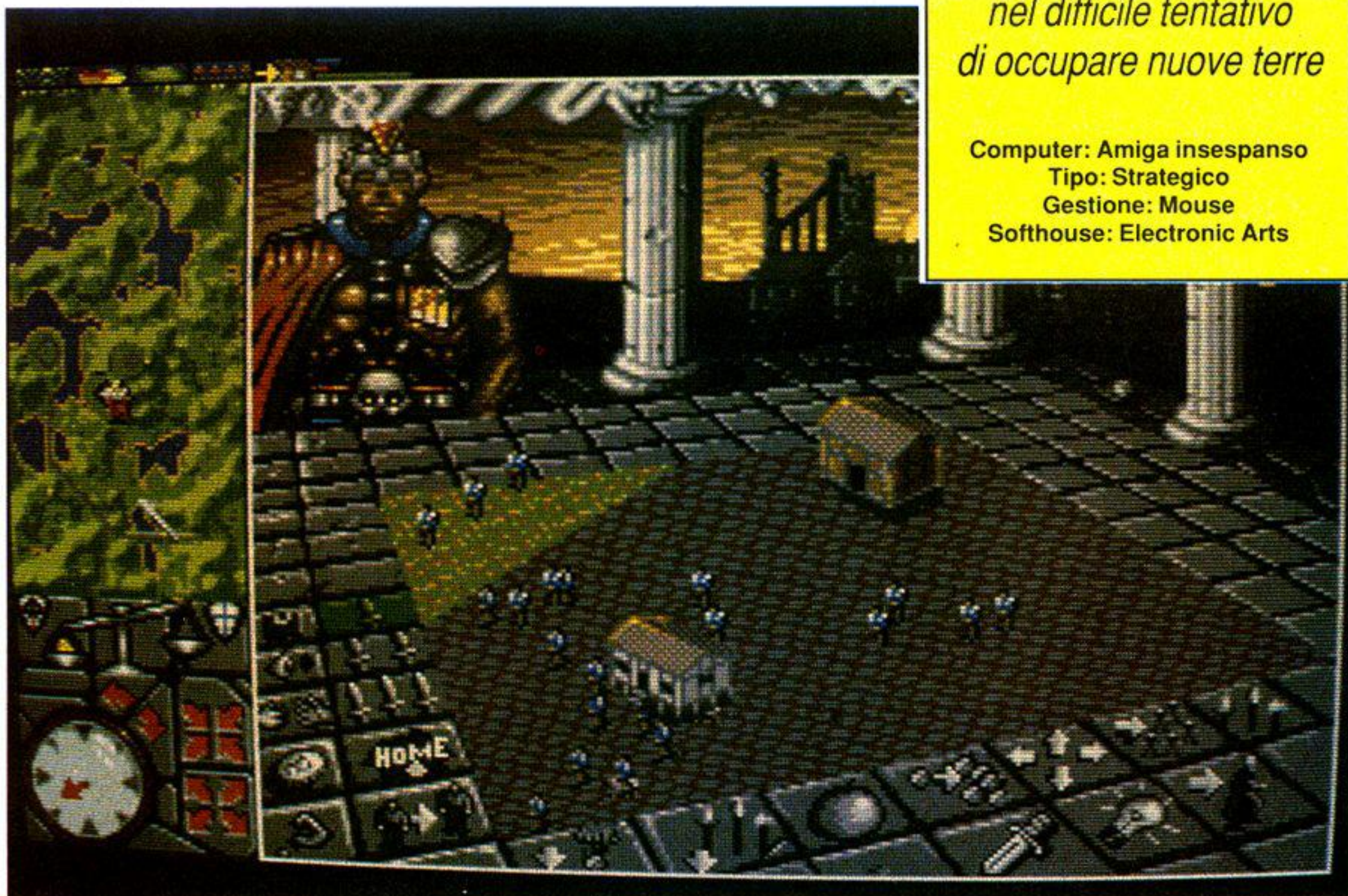
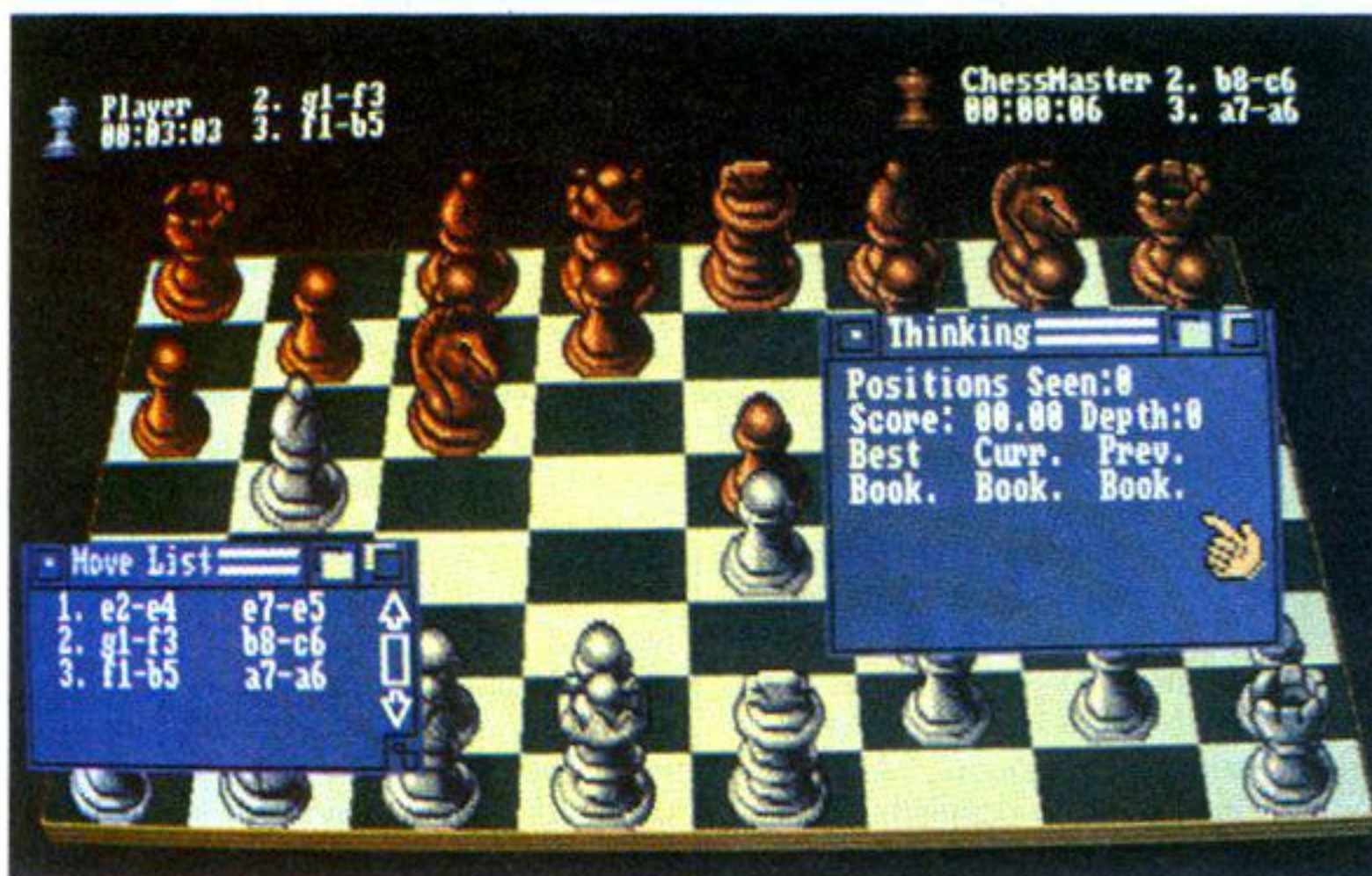
POWER MONGER

Non si tratta del ventilato seguito del mitico **Populous**, sebbene gli autori siano gli stessi, come anche la originalità e l'ottima qualità.

Il gioco

Si vestono i panni di un guerriero esiliato, con il proprio esercitino di fedelissimi, costituito purtroppo soltanto da venti uomini. Lo scòpo è quello di conquistare ed appropriarsi di un nuovo territorio, in cui si è appena sbarcati. Per farlo, si possono controllare singolarmente gli uomini del proprio esercito con appositi comandi. A differenza di Populous, ogni omino ha una grande varietà di azioni "personali", che compie di propria iniziativa se non gli si ordina diversamente.

Passerete moltissime ore alla tastiera, nel difficile tentativo di occupare nuove terre

Computer: Amiga insespanso
Tipo: Strategico
Gestione: Mouse
Softhouse: Electronic Arts

CHESSMASTER 2100

Il nuovo discendente di uno dei primissimi giochi per l'**Amiga 1000** ha delle caratteristiche interessanti, anche se non eclatanti.

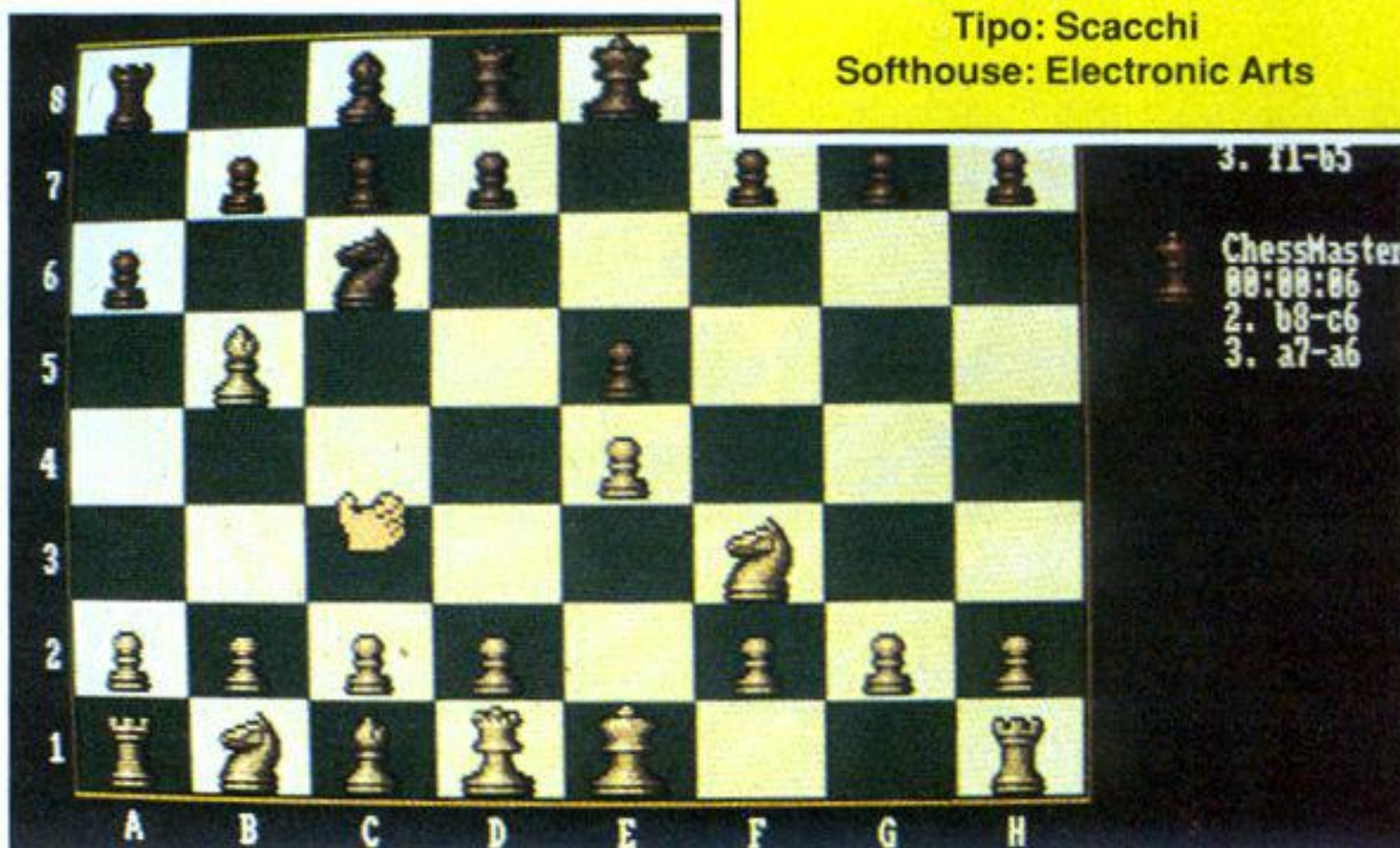
Il gioco

I programmi di scacchi per Amiga più noti sono almeno cinque: **ChessMaster 2000**, **Chess 2175**, **Sargon III**, **BattleChess** ed **Art of Chess.**

Il primo fu, però, qualcosa di unico: un programma ricco di opzioni interessanti e dalla grafica estremamente curata, un vero termine di paragone per tutti gli altri che, sebbene "battessero" poi il ChessMaster originale, presentavano sempre nei suoi confronti delle pecche di vario tipo.

ChessMaster 2100 è nato per migliorare il poco di migliorabile del predecessore, senza esagerare nelle innovazioni. Ora sono previsti ben **due dischetti** ed il programma conosce a menadito centinaia di **varianti di apertura**, che cita e visualizza anche durante la partita.

Inoltre dispone di appositi moduli separati che giocano in modo differente a seconda che ci si trovi in apertura, in medio gioco od in finale, come vogliono gli algoritmi più recenti.

Non mancano, ovviamente, le raffinatezze: vari tipi di pezzi e di colori, possibilità di rotare la scacchiera, di visualizzare le mosse "pensate" dall'elaboratore, di scegliere vari effetti sonori o la sintesi vocale. Purtroppo non è consentito modificare il tipo di gioco del computer (aggressivo o difensivo, ad esempio), come per altri programmi, ma ciò è da imputare probabilmente ad una voluta bilanciatura dell'algoritmo di gioco, che deve semplicemente giocare per vincere o pattare, non per sbilanciare i propri parametri di valutazione in funzione del gioco voluto dall'utente.

La tecnica

La grafica è ineccepibile e gli effetti sonori ottimi, per quanto consente un gioco di questo tipo. L'utente ha parecchie opzioni di regolazione che gli consentono di personalizzare a piacere il gioco ed i livelli di abilità del programma. L'algoritmo di gioco è stato potenziato ed ora è ancora più "imbattibile", specie per la media dei giocatori umani!

Il voto

Il più completo dei programmi di scacchi per Amiga. 9

Chi vuole un avversario forte e privo di scrupoli non può fare a meno di questo gioco

Computer: Amiga inespanso
Gestione: Mouse
Tipo: Scacchi
Softhouse: Electronic Arts

di Luigi Callegari

COMIC SETTER

Volete provare a disegnare fumetti, nuvoletta compresa? Bene, Comic Setter è proprio quello che cercate

Comic Setter è un altro celeberrimo pacchetto per Amiga, **interamente tradotto in lingua italiana** sia per quanto riguarda il **software** sia per quanto riguarda la **documentazione**. La **Leader**, famosa casa italiana distributrice di software per tutti i tipi di personal computer di larga diffusione, lo offre su vasta scala in tutta Italia, grazie ad una rete capillare di negozi.

Il programma, sviluppato in linguaggio **C** ed **Assembler** dalla Gold Disk sulle tracce dei famosi **Professional Draw** e **Professional Page**, è peraltro decisamente potente, originale e semplice da utilizzare, grazie ad un uso efficace ed intelligente delle possibilità di Intuition e dell'interfaccia utente. Merita quindi senz'altro una recensione approfondita.

ComicSetter, gemello del **MovieSetter** visto sul numero **81** di C.C.C. serve essenzialmente per la realizzazione di fumetti, disegnati in proprio, tramite una serie di strumenti e meccanismi che consentono di lavorare come negli studi professionali. L'unica dote veramente necessaria è il **saper disegnare**, altrimenti si finirà sempre con il riciclare il limitato numero di sagome fornite sul disco dati con il pacchetto!

Primi passi

Il programma è dotato di una protezione semplice ed efficace: richiede l'immissione di una parola scelta a caso nel manuale italiano originale, la qual cosa consente, tra l'altro, l'installazione su Hard Disk del programma senza alcun problema. Si può avviarlo indifferentemente da **Shell** o da **Workbench**, tramite apposita icona.

Prima dell'inserimento della **password**, il programma verifica che sia collegata ed accesa la stampante selezionata dall'ultima configurazione e segnala

eventuali problemi al riguardo, usando come default il driver compatibile Epson.

Pagine

ComicSetter funziona per pagine, ovviamente. Sul lato sinistro dello schermo troviamo i comandi del programma e nell'angolo in alto a destra gli indicatori di coordinate, unità di misura (pollici o centimetri). Scegliendo "Aggiungi Pagina" si può appunto aggiungerne una nuova, definita tramite un requester di Intuition che consente di specificarne i parametri caratteristici. Il formato della pagina può essere il classico **A4** (21 x 29.7 cm.) od interamente personalizzato, arrivando in questo caso ad un massimo di **1008 x 1008 pixel** per pagina.

Nel requester di definizione della finestra si possono anche regolare i margini della pagina: da mezzo pollice a zero. Inoltre è possibile, tramite l'opzione **AutoPanello**, creare una pagina con i "pannelli" precostruiti, ovvero le finestre di una classica pagina a fumetti: qui si devono definire il numero di pannelli e la quantità di spazio da lasciare tra ciascuno di essi.

La pagina, come nell'altro pacchetto "Professional Draw" della Gold Disk, può essere ingrandita o rimpicciolita a piacimento (dal 50% al 200%) per consentire un lavoro di precisione, oppure una visione globale dell'insieme come se si osservasse il lavoro "dall'alto".

Pannelli

Per riempire una pagina dobbiamo usare dei "pannelli", che non sono altro che **vignette** rettangolari da posizionare dove si vuole all'interno della pagina. La creazione di un pannello può avvenire in due modi: con il **mouse** (dopo avere cliccato sul gadget) o da **tastiera** (inserendo le coordinate e le dimensioni del pannello). Le dimensioni ed il numero di pannelli per pagina è completamente libero, sebbene possano essere soltanto di forma **rettangolare**. Per la loro collocazione si può utilizzare una classica "griglia", alla Deluxe Paint. Le dimensioni della griglia di riferimento possono venire definite da Preferences, mentre con un'altra opzione si possono muovere e dimensionare gli oggetti rimanendo legati alle linee che formano la griglia stessa.

Una volta creato un pannello è comunque possibile in ogni momento cambiarne le dimensioni e la posizione, eventualmente insieme alle dimensioni ed ai colori del bordo.

ComicSetter consente di usare **sedici colori** contemporaneamente, sia in alta risoluzione (interlacciata) sia in bassa risoluzione. I modi grafici sono comunque quelli standard: niente HAM od Extra Halfbrite, dunque.

Una schermata di Movie Setter

Della stessa s/w house Goldel Disk è anche il programma Movie Setter, presentato sul numero scorso, che consente di realizzare animazioni grafiche con la massima semplicità.

Grafica

La grafica di ogni pannello può essere di due tipi: **bitmapped** o **strutturata**, come scelto dalla opzione **Modo Grafico**. La prima è la classica grafica generata da programmi tipo Deluxe Paint, con una paletta di sedici colori, più dispendiosa in termini di consumo di memoria.

La grafica strutturata è invece più compatta, ma deve essere tracciata **vettorialmente**, la qual cosa le consente di rimanere immutata durante le variazioni di dimensioni. La grafica strutturata può essere tracciata con un numero limitato di strumenti di disegno: mano libera, linee continue, spezzate, cerchi, ellissi, rettangoli e curve di Bezier. Ovviamente, di tutti si può indicare lo spessore della linea e le sagome di riempimento; ComicSetter fornisce ben 25 maschere ("pat-

tern") standard di tracciatura, che però non possono essere ridefinite.

La grafica bitmapped consente più operazioni: importare pannelli creati da Deluxe Paint, ad esempio, o da programmi che abbiano comunque l'uscita grafica in formato **IFF**. Il disegno a mano libera, già dall'interno di ComicSetter, può essere fatto con quindici tipi di pennello, uno spray e si possono mescolare due colori già presenti per ottenere sfumature. Tra l'altro, nel disco dati fornito sono presenti varie figure IFF di supereroi e scene varie che possono consentirci di muovere i primi passi col programma senza troppi problemi.

Fumetti

Dopo avere creato una vignetta, con sfondi e personaggi ben disegnati e collocati, è possibile inserire dei fumetti, o "nuvolette" (o pipe) che dir si voglia. Per farlo, basta cliccare sul gadget a forma di fumetto, appunto, contenente una lettera **B**. Il programma consentirà di immettere del testo in vari formati e stili, secondo le ben note capacità grafiche di Intuition.

Sono previsti sei diversi tipi di fumetto: cliccando due volte sul gadget appena descritto viene aperta una finestra in cui si può scegliere dal classico tipo a nuvoletta ad altre forme geometricamente più o meno regolari.

Alcuni comandi, riportati sulla parte destra della finestra, sono attivi soltanto su particolari tipi di fumetto e consentono, ad esempio, di segmentare la pipa in modo da indicare, ad esempio, il "pensiero" del personaggio. Anche la **coda** di collegamento tra il fumetto ed il personaggio può essere regolata comodamente, tramite valori numerici o gadget.

Stampa

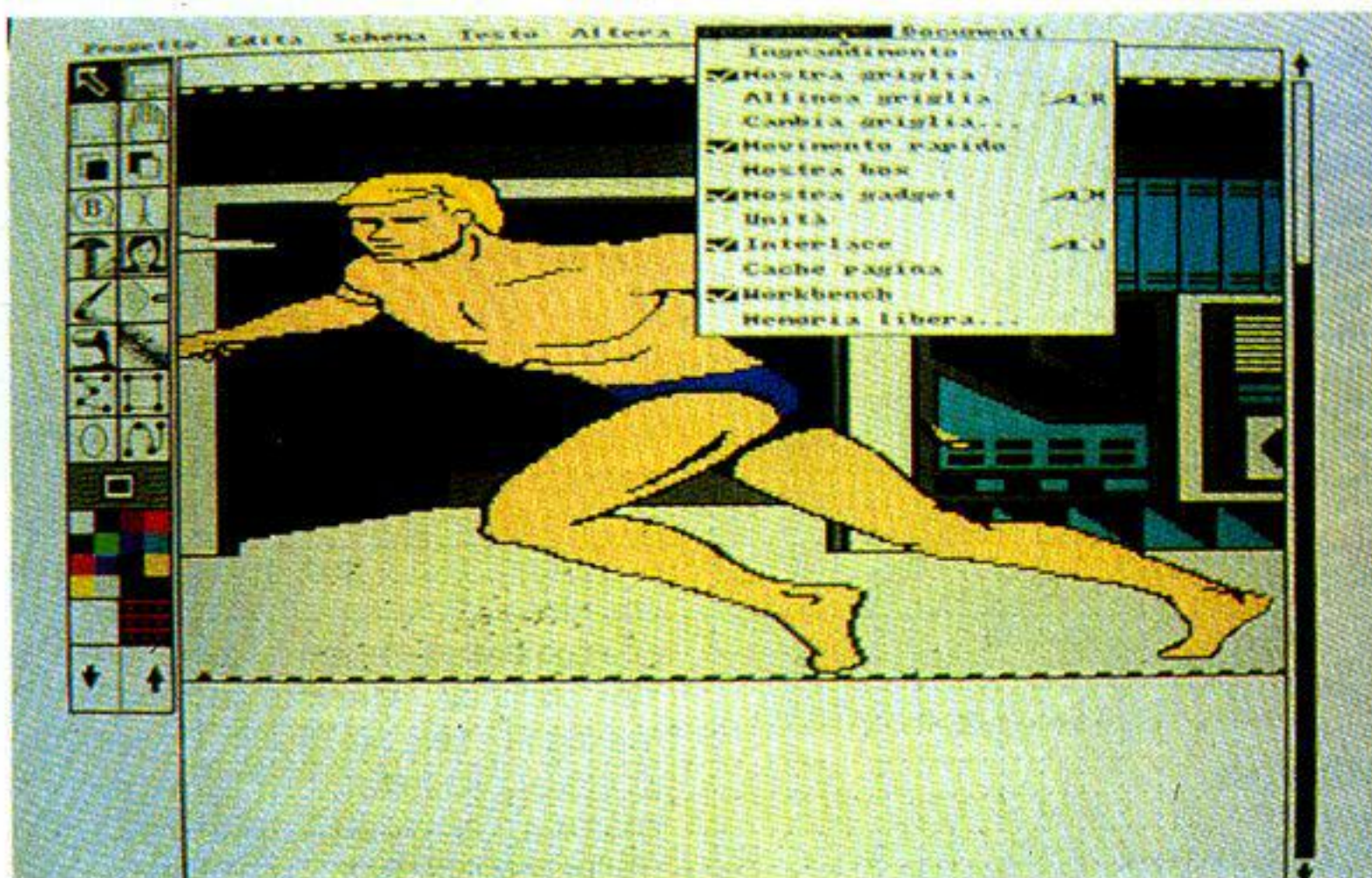
Tutto il nostro lavoro di produzione varrebbe a poco se poi non lo si potesse stampare efficacemente. ComicSetter adotta un sistema particolare di interfaccia stampa: ciò che si vede sul video **non** è esattamente ciò che verrà stampato (non WYSIWYG), ma la stampa sarà comunque di altissima qualità, una volta abituato l'occhio a vedere le cose come le vede il programma.

La stampante deve essere correttamente configurata da **Preferences** ed i parametri di stampa grafica, come ad esempio la densità in punti per pollice, devono venire scrupolosamente immessi tramite il requester di "Regolazione ambiente". La qual cosa consente di usare anche stampanti che non sono presenti nell'elenco di driver supportati direttamente dalla Commodore per Amiga, sfruttandole al meglio delle loro capacità grafiche.

Conclusioni

Il programma ComicSetter è certamente molto interessante per tutti coloro i quali desiderano cimentarsi in un lavoro creativo come quello della produzione di fumetti o, comunque, creazioni grafiche analoghe. La produzione di un giornalino o di un bollettino stile fumetto può essere estremamente semplice e gradevole con questo programma, ottimamente curato sia nella veste grafica sia nelle traduzioni del manuale e del software.

A riprova della serietà con la quale è stato prodotto, diciamo che la Gold Disk ha già rilasciato una **dozzina** di dischetti contenenti sagome e dati grafici adatti ad un rapido inserimento in proprie creazioni.

Il prezzo, di circa **70.000 lire**, ci sembra più che giustificato per un programma con queste caratteristiche.

di Domenico Pavone

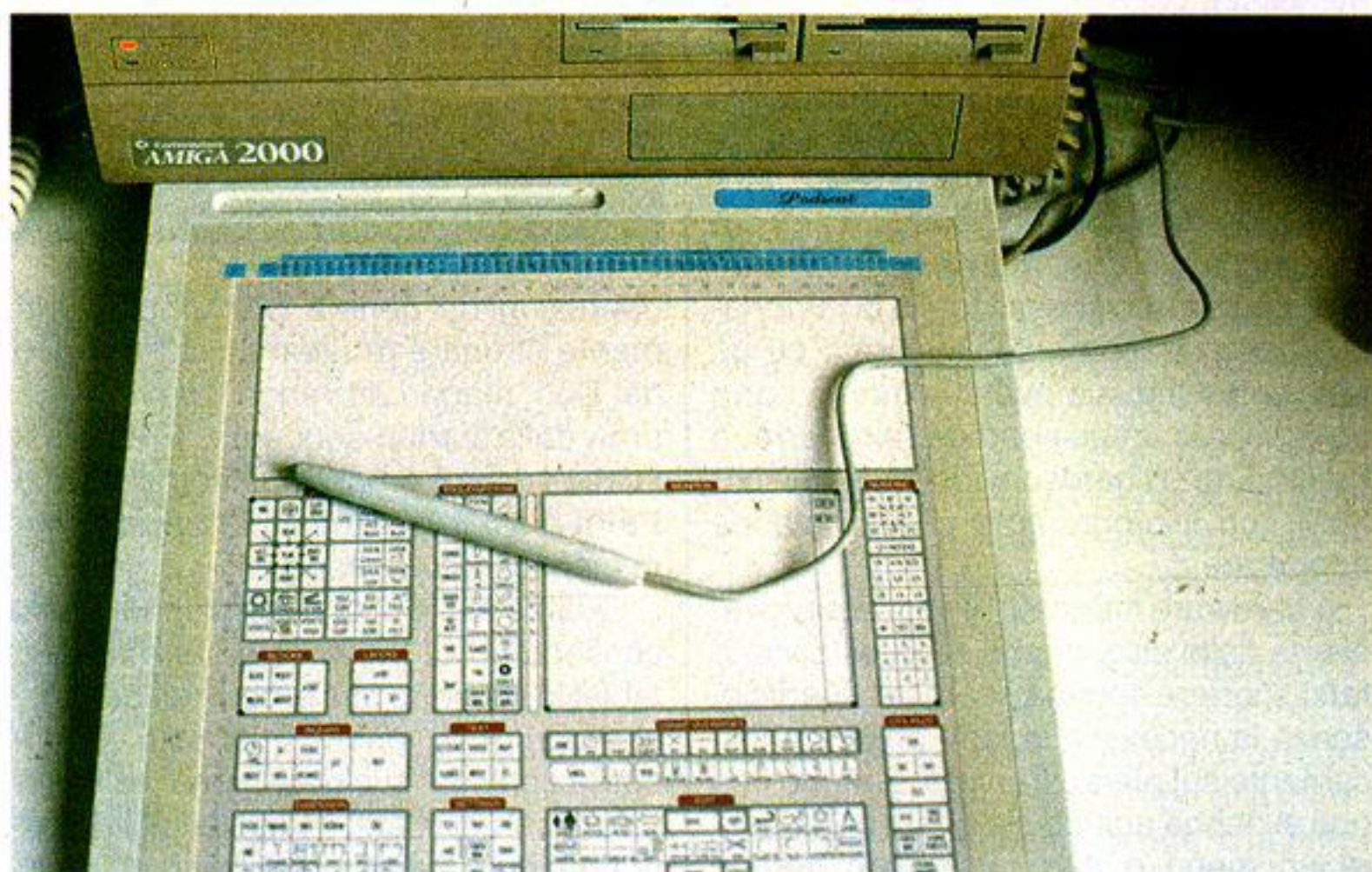
TRE NOVITA' PER AMIGA

Chi ha sete di accessori troverà, questo mese, di che saziarsi...

Amidraw tablet

Il package consiste, di fatto, in **due** elementi costitutivi: uno **hardware** ed uno **software**. Tale premessa, in apparenza scontata, rende invece più appetibile di quanto si pensi una periferica di per sè già molto interessante, come si capirà tra breve.

La **Podscat pt3030** è una tavola grafica, ovvero uno strumento votato alle applicazioni grafiche, in grado di facilitare al massimo l'input di qualsiasi programma del genere. Detto in termini più pratici: si sostituisce al mouse, consentendo il tracciamento effettivo di disegni, grafici e similari esattamente come se si stesse operando con una più familiare "periferica" fatta di carta e penna.

Definirla in questi termini, è comunque alquanto riduttivo. L'accessorio, in realtà, è in grado di fare molto di più, e non solo in ambiente Amiga!

L'hardware possiede, infatti, il pregio di essere adattabile anche alle caratteristiche dei **PC** compatibili, per non dire soprattutto: nasce, cioè, evidentemente rivolto alle applicazioni in **Cad** del mondo Ms-Dos, garantendo una pressoché totale emulazione dei vari "mouse standard" di quell'ambiente, tanto che viene fornito con un foglio-guida per facilitarne l'uso con **AutoCad**.

Tutto ciò, comunque, non ne rende meno valido l'uso su Amiga, tutt'altro.

La tavola, di dimensioni non certo ridotte (cm. 40 x 40) a garantirne il massimo della comodità, va collegata alla porta seriale del computer (**Pc** o **Amiga** che sia) mediante un cavo fornito in dotazione, il che mantiene agibile in contemporanea il mouse, utilizzabile quindi per altri scopi. L'alimentazione è autonoma, grazie ad un (piccolo) alimentatore da inserire direttamente in una presa di corrente.

Alla tablet, come ovvio, è associato uno **stilo**, che svolgerà le mansioni del mouse, impugnabile come un qualunque altro strumento di scrittura (penna, pennello, etc.). Come ovvio, con qualche

Che bolle in pentola

La vetrina hardware di questo mese si occupa di tre soluzioni hardware decisamente diverse tra loro: una **Trackball** da sostituire al mouse, una **tavoletta grafica** per un approccio più confortevole ed abituale al mondo del disegno semi-professionale su Amiga, ed infine un **emulatore Pc-At**, la ATonce.

Molta carne sul fuoco, si penserà, ed in parte a ragione, ma di fatto solo due numeri fa si è già recensita una scheda hardware di emulazione **Pc-Xt** per Amiga 500. Ovvio, quindi, esaminare l'ultimo elemento più che altro in termini di raffronto con la precedente, piuttosto che approfondirne solo le caratteristiche generali.

Un po' di "quantità" in fondo non guasta e, a dispetto della proverbiale antitesi, non necessariamente a discapito della qualità.

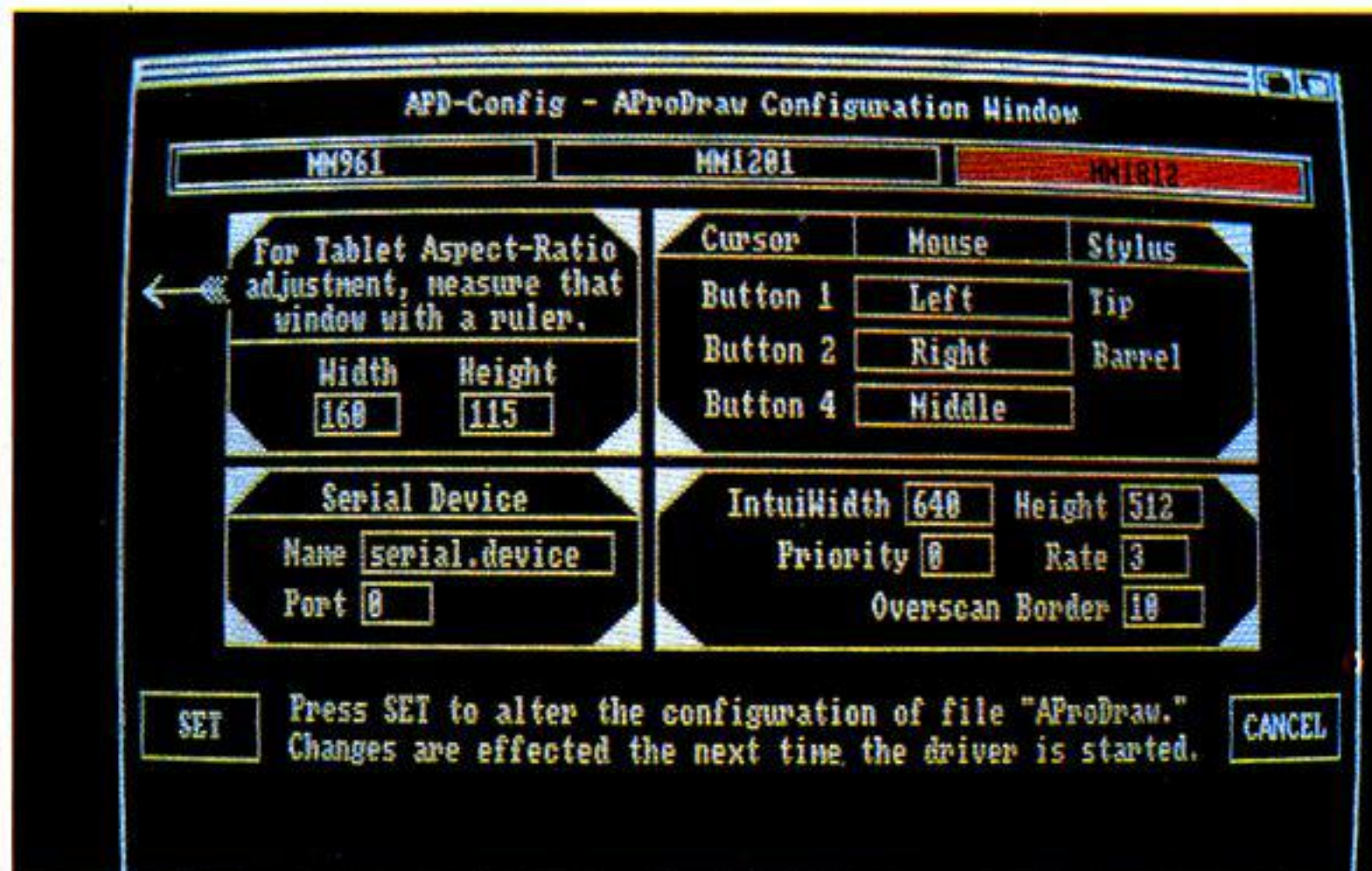
La sete di informazione sull'hardware è sempre in agguato, soprattutto quando si ha a che fare con gli altrettanto proverbiali (bassi) istinti del popolo "amigo".

differenza: dovendo "disbrigare" tutti i compiti comunemente riservati al mouse, è fornito di due piccoli pulsanti nella parte bassa dell'impugnatura, e di una punta rientrante.

Quest'ultima, con una semplice pressione della "penna" sul piano di lavoro, emula l'effetto del pulsante sinistro del mouse, così come uno dei due pulsantini sul corpo dello stilo. L'altro attiva, invece, le funzioni di menu di Amiga, esattamente come se si adoperasse l'equivalente pulsante destro del mouse. Questa configurazione di default può, però, essere modificata via software come più aggrada, assegnando a ciascuno degli elementi (compresa la punta rientrante!) la funzione che risulta più comoda per le proprie applicazioni. Prima di esaminare l'indispensabile corredo software, va ancora aggiunto che, sul ripiano della tablet, poggia uno spesso foglio di materiale plastico trasparente, ancorato su un lato, sotto al quale eventualmente porre un grafico o un disegno da riprodurre, o (perché no?) l'hardcopy dei menu disponibili con il programma che si intende adoperare.

L'hardware intercetta il segnale proveniente dallo stilo, anche se questo viene fatto scorrere appena sopra la superficie, senza la necessità di farlo poggiare fisicamente sul piano di lavoro, a meno che non si debba agire sulla punta per attivare un menu o altro. Per l'uso vero e proprio, è comunque indispensabile qualche manovra software.

Allo scopo e' accluso, al package, un floppy contenente un "driver" per la tablet, nonché due programmi di configurazione. Vedere all'opera lo stilo è più semplice di quanto si creda: basta un normalissimo doppio click da ambiente Workbench sull'icona **AProDraw** (che sta per **A**miga **Pro**fessional **Draw**), mentre la stessa operazione sull'icona "Aprodraw" disattiverà il driver. Dopo lo start, si può subito constatare come il cursore sullo schermo seguirà ogni movimento dello stilo, purché questo venga fatto scorrere sulla superficie della tavola.

Per un uso ottimale, può risultare più comodo copiare i due files appena citati nel disco contenente l'applicativo che si intende adoperare; operazione facilitata dall'assenza di protezioni nel dischetto, e dalla possibilità di eseguirla direttamente da Workbench, manovrando le icone come di consueto. I più avvezzi a

operazioni del genere, potranno naturalmente sfruttare in alternativa i comandi del Dos, magari attivando il driver Aprodraw dalla startup-sequence. In entrambi i casi, l'uso di programmi come **Deluxe Paint** o similari, cambierà radicalmente. In meglio, naturalmente.

Quanto ai file di configurazione, questi consentono di modificare l'attività della tablet in rapporto alle proprie esigenze, o ai propri gusti. Anche in questo caso le operazioni da svolgere sono estremamente semplificate, e tutte gestibili tramite riquadri e icone molto chiari ed esaurienti, adoperando il mouse o lo stilo, se già attivo. In particolare, con **ADP_Config** si potrà settare quella che diverrà la condizione di default della tavola in rapporto alle dimensioni dello schermo ed alla sua taratura fisica, soprattutto nel caso di monitor diversi da quelli standard Amiga. Come già accennato, è poi possibile assegnare una funzione diversa da quella predeterminata a ciascun pulsante della penna ottica, nonché cambiare, se il caso, il nome del Serial.device adoperato dal sistema o la priorità del driver software, eventualmente da sperimentare in rapporto al programma grafico sfruttato.

Ancora più utile e versatile il secondo file, **ADP_Scale**, i cui settaggi possono, in talune circostanze, risultare decisivi. Anche in questo caso, con difficoltà pressoché nulla, è possibile variare il rapporto tra lo scorrimento dello stilo sulla tavola e l'effettivo movimento del cursore sullo schermo. Inizialmente, per default, tutta la tavoletta rappresenta, in pratica, l'intero schermo. Quasi sempre è però più comodo far sì che solo una parte della tablet corrisponda al video, sia per rendere possibili riproduzioni "a pantografo", sia per una praticità legata ai movimenti della mano.

Ebbene, con Adp_Scale si può variare questo rapporto come più aggrada. Selezionato l'opportuno riquadro, per esempio, basta pressare la punta della "penna" su quello che sarà il margine superiore sinistro della porzione di tablet desiderata, e quindi ripetere l'operazione sul margine inferiore destro. Tutto fatto. Per un uso normale, può essere utilizzato lo stesso foglio fornito in dotazione, pur se dedicato ad AutoCad. Questo riporta infatti una porzione "monitor" di dimensioni ridotte, con su un lato una banda riservata ai menu di schermo. Adoperando questa sezione per intero, avremo (p. es.) proprio la stessa conformazione adoperata da Deluxe Paint, per cui si potrà molto agevolmente non solo disegnare, ma anche selezionare le varie opzioni del programma poste lateralmente sul video.

Cos'altro aggiungere, se non un giudizio più che positivo? Chi fosse già troppo abituato all'uso del mouse, potrebbe all'inizio trovare qualche difficoltà di "ambientamento" nel tornare a qualcosa in fondo già familiare come una penna tra le dita, ma è solo il primo impatto. In fondo, chi ha già imparato a nuotare, resterà a galla anche dopo anni che non vede il mare o una piscina.

E ad usare carta e pena, si presume, abbiamo imparato tutti.

Amtrac

Sebbene il mouse, per un utente di Amiga, diventi presto un'appendice pressoché naturale della mano (o quasi...), non è in effetti l'unica alternativa disponibile per svolgere i compiti cui siamo tutti abituati. Particolari esigenze di precisione, o anche solo problemi di spazio, possono in certi casi consigliare di avere a portata di mano una cosiddetta **trackball**, ovvero una periferica per l'input che fisicamente funziona in modo inverso al mouse: invece di far ruotare una sfera muovendo il supporto che la contiene (la pallina del mouse, per intenderci), si opera direttamente sulla sfera, mentre il supporto rimane fermo.

E' in questa categoria che rientra AmTrac, un prodotto della statunitense Microspeed Incorporated. Si tratta di un oggetto dalle dimensioni più voluminose di un normale mouse, come d'altronde necessario per uno strumento che non solo deve rimanere ben ancorato al piano di lavoro, ma anche consentire un comodo appoggio della mano su di esso.

L'installazione non presenta alcuna difficoltà: basta inserire il connettore nella porta comunemente usata per il mouse, e la trackball è immediatamente attiva.

Sulla sua superficie superiore spicca naturalmente la grossa sfera adibita al movimento del cursore, cui ci si abitua con molta rapidità. Il meccanismo risulta estremamente fluido, e va aggiunto, a suo merito, che l'elemento più sollecitato (la sfera) è costruito con un materiale plastico autolubrificante, che molto di rado necessita di manutenzione (pulizia, per in tenderci).

Ai lati della "ball", due vaste sezioni della superficie sono occupate dai pulsanti, la cui disposizione e dimensione ne consente un utilizzo ottimale quale che sia la posizione della mano sulla sfera. Al centro, anteriormente, è poi presente un terzo pulsante, che consente di facilitare al massimo una tipica manovra del mouse, che con le trackball può diventare più complessa: implementare il movimento del pointer sullo schermo in contemporanea alla pressione di un suo pulsante. Per intenderci, il tipo di azione necessaria per ridimensionare una finestra di Intuition, o disegnare con un programma grafico.

Con AmTrac, tutto ciò che si deve fare è premere e rilasciare il pulsante centrale. Fino alla successiva pressione dello stesso, il movimento della sfera resterà associato alla pressione del pulsante sinistro, senza la necessità di mantenerlo premuto fisicamente. E non è finita qui. La stessa azione è applicabile anche al tasto destro, consentendo per esempio di scorrere e soffermarsi molto comodamente tra i menu della title bar dello schermo limitandosi ad agire sulla sfera. In questo caso, basterà premere (sempre una sola volta, s'intende) il pulsante centrale mentre è abbassato quello destro, e quest'ultimo resterà attivo fino a che non si agirà nuovamente su di esso o su quello centrale.

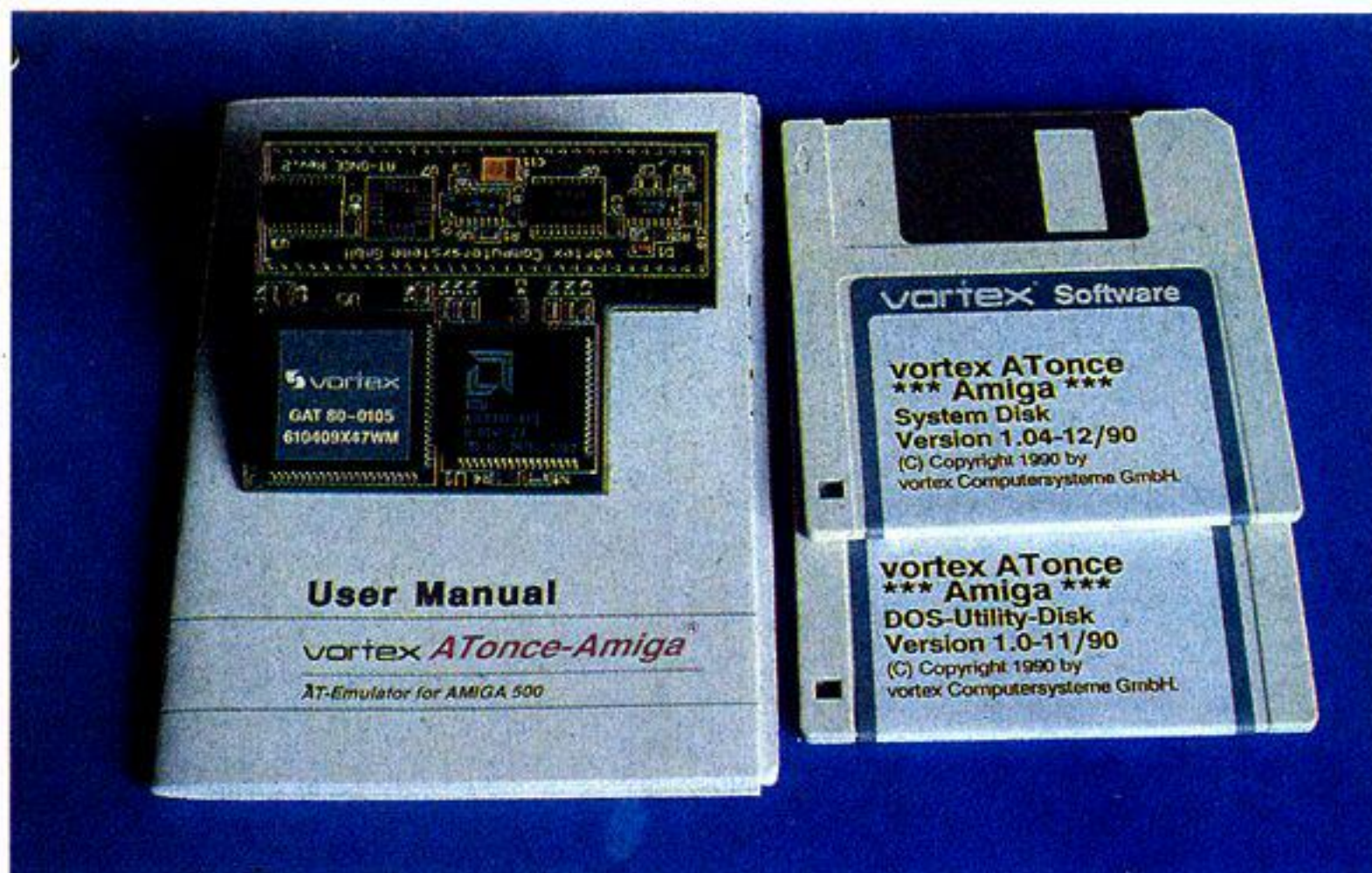
AmTrac può, insomma, essere definito come un optional che, una volta provato, rischia di soppiantare il mouse anche nell'uso di tutti i giorni, games compresi.

ATonce Emulator

Non si è ancora spenta l'eco della comparsa sul mercato di un emulatore **Pc-Xt per Amiga 500** (la Pc PowerBoard, recensita nel n. 80 della rivista), che un nuovo subbuglio Ms-Dossiano ha cominciato a far scalpitare i possessori del "piccolo" 500.

Il motivo di tale sommovimento? Ma la ATonce, naturalmente. Il prodotto della Vortex, distribuito per l'Italia dalla **Flopperia di Milano**, si è presentato prepotentemente sul mercato con qualche carta in più rispetto alla già citata PcBoard, vuoi per il prezzo notevolmente inferiore, vuoi per l'utilizzo del microprocessore **80286**, appartenente notoriamente alla categoria dei "16 bit".

E' bene precisare subito, prima di approfondire l'esame della scheda, che comunque un paragone diretto tra i due emulatori hardware non è in realtà così immediato, e ognuna delle due presenta aspetti che possono, in positivo o in negativo, far decidere sull'acquisto di una piuttosto che l'altra. Ma procediamo con ordine.

La prima cosa che colpisce della ATonce, è sicuramente la ridottissima dimensione: come visibile nelle foto, praticamente è meno ingombrante di un floppy da 3.5! L'installazione fisica, a differenza della PcBoard, va eseguita a "macchina aperta", peraltro con estrema semplicità, affidandosi alle esaurienti foto mostrate nel manualetto in dotazione.

Breve parentesi: oltre al manuale ufficiale (in inglese), quando leggerete queste righe ne sarà quasi sicuramente già disponibile la **versione italiana** curata dallo stesso distributore (Flopperia), che per più non si limiterà ad una semplice traduzione, ma aggiungerà qualche "trucco" per facilitare l'uso della scheda.

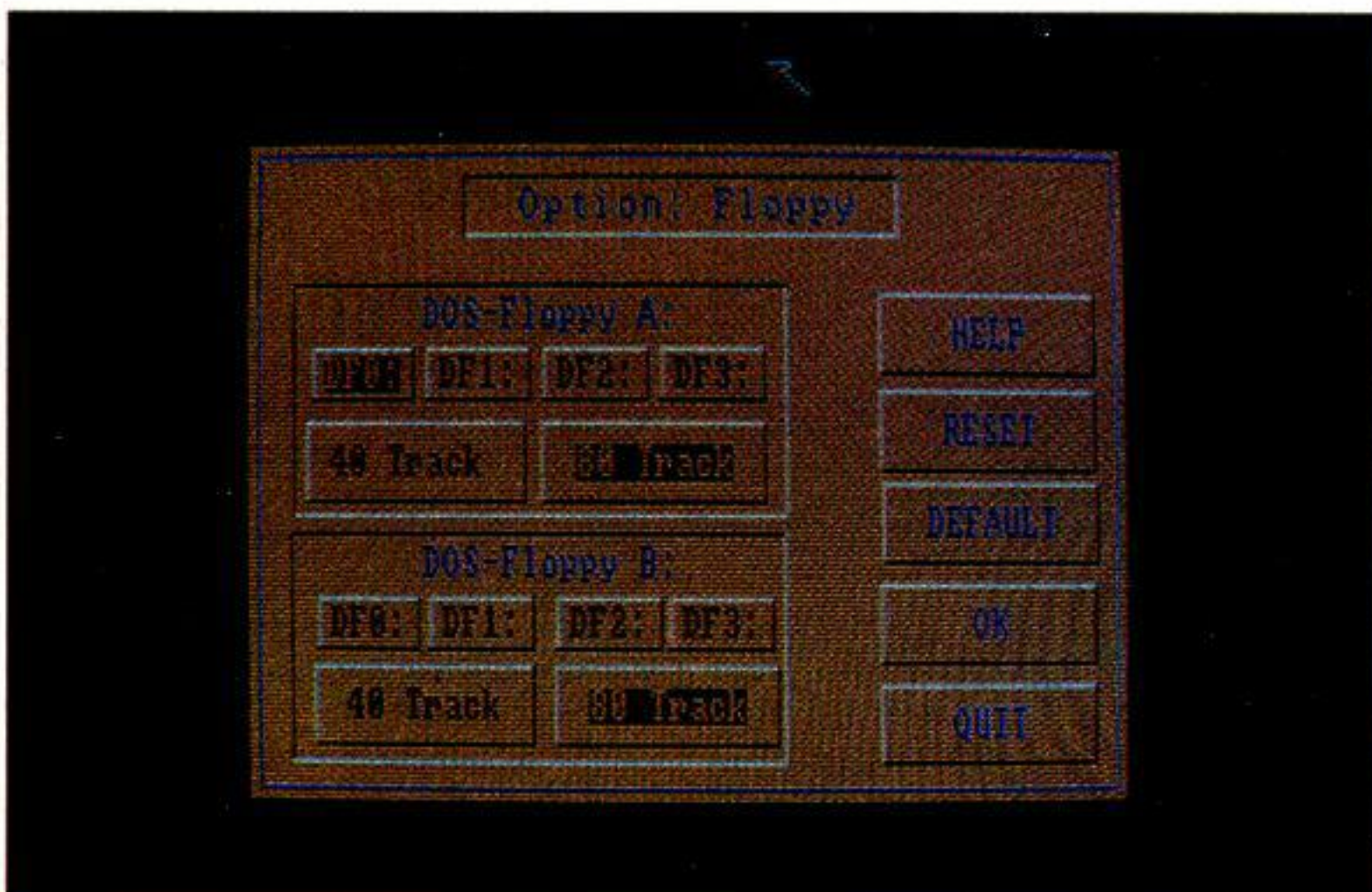
Chiusa la parentesi, torniamo all'installazione, che prevede in poche parole il disinserimento del 68000 (basta un cacciavite per far leva, niente di complicato), al posto del quale va infilato quello presente nella scheda. Il "vecchio" 68000, può essere riposto in una bustina antistatica fornita assieme alla ATonce.

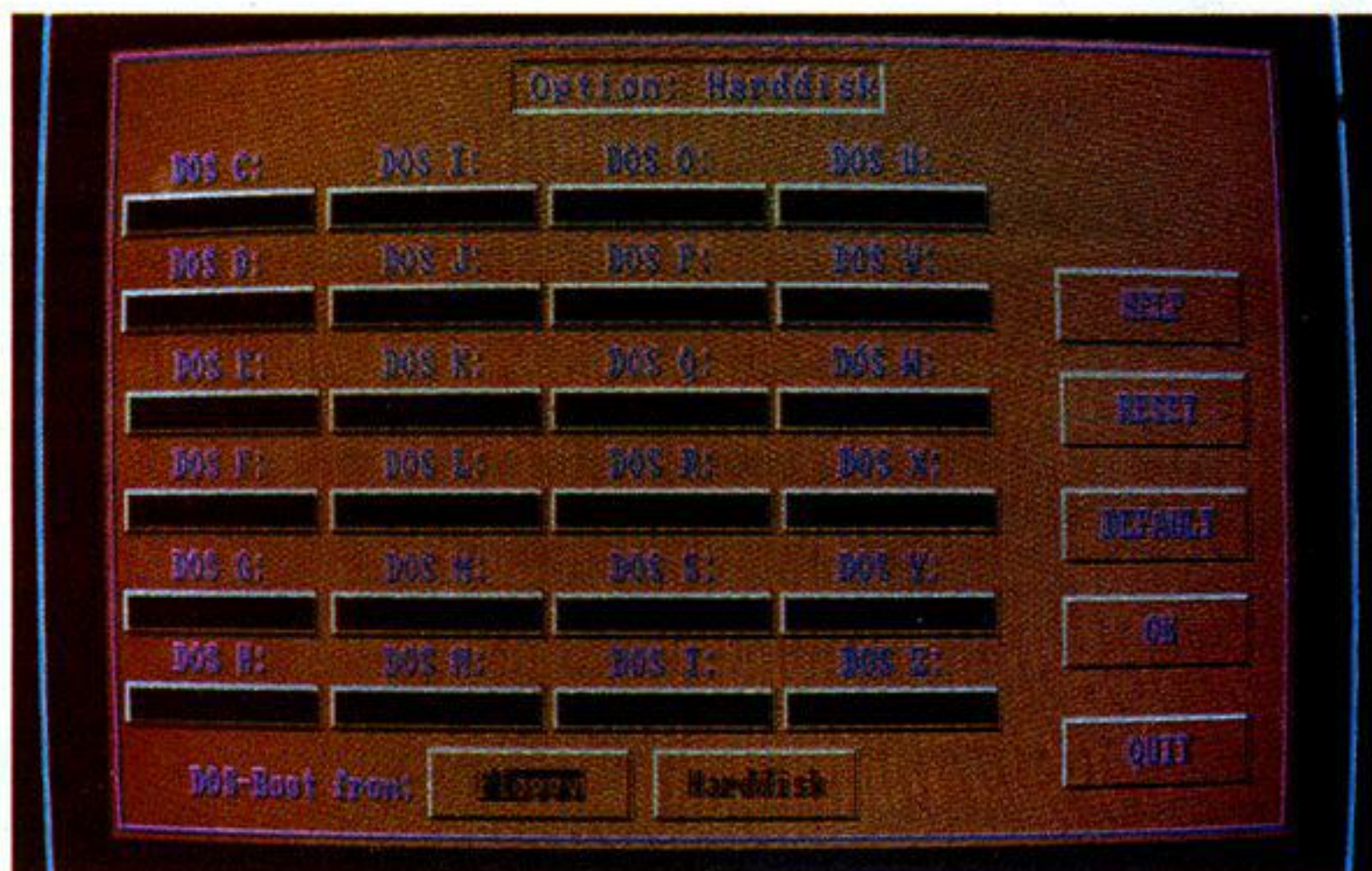
In alcune configurazioni, specificate tanto nel manuale quanto in un file "readme" che viene mostrato allo start del disco di sistema a corredo della scheda, può anche rendersi necessario inserire un controzoccolo sotto il chip Gary (anche questo molto facilmente), più che altro per evitare rallentamenti nell'uso in modo Amiga.

Già in questa breve introduzione si può cogliere la prima notevole differenza con la scheda Xt, che invece si installava direttamente nel cassetto inferiore di Amiga 500. Il vantaggio della ATonce, in questo caso, si manifesta soprattutto per chi già disponesse di una qualche espansione ivi installata, o comunque intendesse fornirsene.

E già che si è in tema di memoria, emerge subito un'alta differenza tra i due modi di emulare l'ambiente Pc. La ATonce, a differenza della PcBoard, **non** dispone di propria Ram, ma sfrutta quella a disposizione su Amiga, anzi adeguandosi a seconda di quanta ne trova. Può cioè conformarsi anche con soli 512 KB, ma raggiunge l'optimum con un mega di Ram, in questo caso adottando il tipico standard del mondo Ms - Dos, che prevede una memoria base di 640 KB. Come sempre, però, più memoria si ha a disposizione, più vantaggi se ne ricavano. Con almeno 1.5 MB, infatti, ATonce può addirittura gestire una Ram estesa o espansa, croce / delizia dei possessori di "veri" Pc.

Altro aspetto da elogiare, la capacità di adattarsi ai drive presenti nella configu-

razione Amiga, compreso un eventuale **hard disk**. Unico particolare al quale occorre abituarsi, un'attesa di 3-4 secondi ogni volta che si cambia disco, tempo necessario per il suo riconoscimento in modo Ms - Dos (cosa peraltro segnalata dalla spia luminosa del drive).

E siamo giunti alla caratteristica forse più interessante di ATonce, che la differenzia nettamente da quanto prima esistente. Il Pc emulator può, per così dire, "condividere" non solo qualche partizione di hard disk con l'ambiente Amiga, ma anche l'attività generale del computer. Anche se allo start di Atonce Amiga si resetta per entrare in modo Ms - Dos, questo risulterà girare in un **task**, mentre (p. es.) dietro, tirando giù lo schermo come di consueto, resta agibile il Workbench. In altre parole, sfrutta il multitasking di Amiga, pur se con le dovute precauzioni del caso. Far girare qualcosa di Amiga, in contemporanea, presenta qualche rischio "guroso", soprattutto in configurazioni minime di memoria. Comunque, resta pur sempre un notevole progresso evolutivo nel settore degli emulatori, hardware o software che siano.

Da un punto di vista pratico, ATonce si attiva molto banalmente con un semplice doppio click sull'omonima icona inserita nel disco di sistema fornito a corredo, contenente anche un file **Install** (anch'esso agibile direttamente da Workbench) da adoperare per settare opportunamente le caratteristiche dell'ambiente Ms - Dos.

ATonce consente infatti l'implementazione di più standard grafici (**CGA**, **Olivetti**, **Hercules**, **T 3100**) a colori o monocromatici con schermo standard o in terlacciato selezionabili tramite il file Install, che consente anche di scegliere se operare il boot in ambiente Pc da floppy o da hard disk, di modificare la tastiera, di precisare quanti e quali floppy driver adoperare (è consentito anche l'uso di drive da 5.25 esterni!), nonché una tradizionale regolazione della Palette colori per decidere quale adottare di default.

Il tutto, naturalmente, adoperando solo il mouse, in modo amighevole che più non si può.

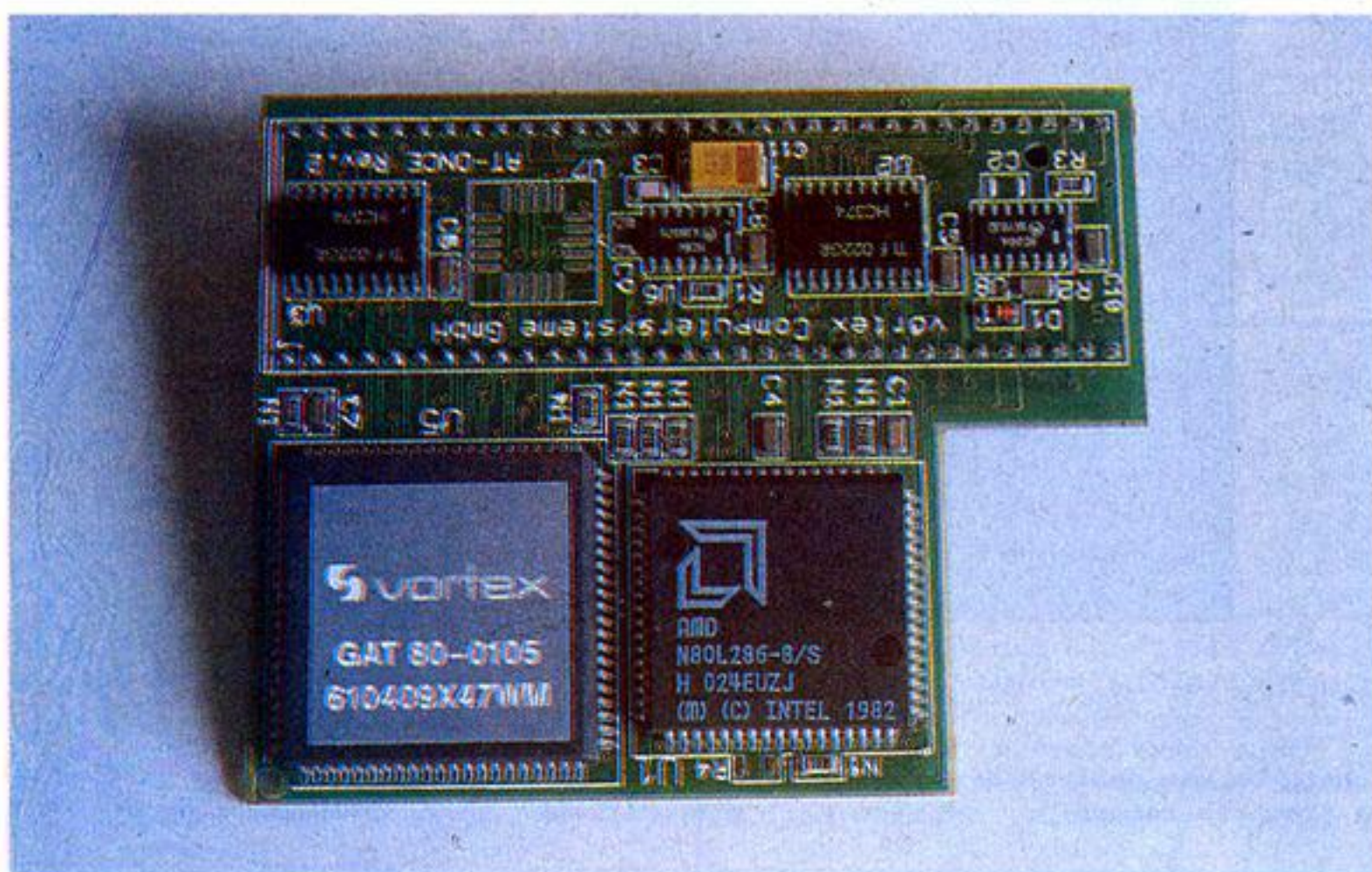
Naturalmente, dopo aver configurato a proprio piacimento quello che sarà l'ambiente Pc, ed attivata ATonce, si dovrà attivare il sistema Pc con un suo disco, ovvero procurarsi una versione dell'Ms-Dos, non compreso nella confezione (d'altronde, per un prezzo così basso...).

Per concludere, non resta che accennare a quanto sicuramente sta più a cuore dell'utente: compatibilità con i programmi Ms - Dos, e velocità di esecuzione.

Quanto alla prima, è decisamente ottima, superiore anche alla PcBoard, consentendo di accedere (memoria permettendo) ad elaborazioni grafiche di tutto rispetto, all'ambiente **Windows**, e così via... *ms-dos*eggiando.

Non bisogna invece aspettarsi miracoli sotto l'aspetto "sprint". Per quanto la scheda sia AT, sfrutta pur sempre il clock di Amiga 500, senza quindi superare i 7.2 Mhertz. Le applicazioni grafiche girano con sufficiente velocità, mentre non altrettanto si può dire del normale scrolling video.

Non bisogna però dimenticare che, per ottenere qualcosa di più, il costo sarebbe sicuramente lievitato, divenendo non più concorrenziale rispetto all'acquisto di un vero Pc da affiancare ad Amiga.

Per un prezzo inferiore alle 500 K lire, il gioco, anzi... ATonce... vale decisamente la candela.

PC TOOLS

Lavorando in ambiente Dos si ha spesso a che fare con i numerosi comandi disponibili (copia, cancellazione, gestione subdirectory, ecc.). Non sempre è agevole ricordare la sintassi da seguire e si rischia di commettere errori di vario tipo, talvolta disastrosi. Sono quindi stati realizzati numerosi Tool che facilitano la gestione dei comandi. Tra questi merita un posto di rilievo **PC Tools** che, molto diffuso tra gli utenti, è una valida "interfaccia" amichevole.

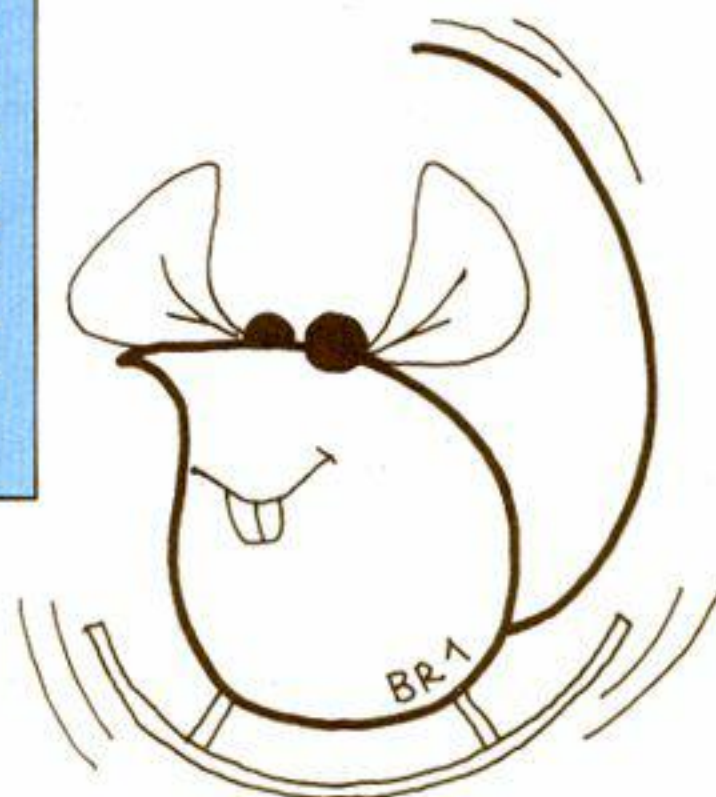
La famosa **"interfaccia utente"** PC Tools è giunta ormai alla sesta edizione. Ciò significa, indirettamente, che le versioni precedenti hanno saturato il mercato degli utenti **Ms-Dos** le cui esigenze, appunto, hanno indotto la nota s/w house ad aggiornare il già potente pacchetto di utility. Moltissimi neo-utenti, quindi, possono facilmente rintracciare una delle precedenti versioni che, ne siamo sicuri, hanno ancora molto da dire, soprattutto a chi si è procurato un computer Ms-Dos da poco tempo.

Schermata di presentazione. Non appena si "lancia" il programma, alcune previ parole di benvenuto precedono le **tre** opzioni selezionabili a questo livello (escludendo il caso banale di **Esc**, che permette l'abbandono di PCT):
F3. La pressione di questo tasto seleziona una delle due principali opzioni, vale a dire la possibilità di operare su caratteristiche che riguardano l'intero dischetto.
F10. Con la pressione di questo tasto si può selezionare, prima di iniziare il lavoro, il dischetto e/o la directory dalla quale operare. Quando, infatti, si fa partire PCT, la directory attiva è quella nella quale ci si trovava al momento del lancio. Si tenga presente, comunque, che in qualsiasi momento è possibile cambiare sia dischetto che directory.
Tasto qualunque. Premendo un qualsiasi altro tasto si entra nel menu che consente, prevalentemente, la manipolazione dei singoli files.

```
WELCOME !

PC Tools Deluxe

(C)Copyright Central Point Software, Inc.
Unauthorized duplication prohibited.

Press any key for File Functions

OR

F3=go directly to Disk and Special Functions

F10=change drive/path from A:\

Press ESC to Exit
```

Schermata di presentazione

```
Path=A:\              BLINK=DOS current

R--GABBIE--------POSTA64
O |              -GAMES
O |              -ARTICOLI
T |              -INDICE
  |              -EDITORIA
  |              -AMIGADOS
  -MOVIESET
  -AMIGAM_1
  -AMIGA_C
  -T_PASCAL
  -HCOPY_64
  -GEOS64
  -POSTA_GE
  -SFIDA_01
  -CONVERTE
  -INVERSIO
  -EDITOR_1
  v

Use cursor control keys to follow the chain to the desired directory.
Press ENTER to accept the choice. Press "Esc" to return.
```

Subdirectory. Prima di consentire l'accesso ad uno dei due menu principali, PCT verifica l'eventuale presenza di subdirectory. In caso affermativo ne visualizza sia la **presenza** sia il **percorso** che le collega tra loro (in caso di "nidificazione" di più directory). La directory "attuale" viene resa lampeggiante; l'utente può decidere di lavorare su directory diverse limitandosi a spostare il cursore sul nome delle stesse e premendo il tasto Return per selezionarle definitivamente.

FILE FUNCTIONS

File Functions. Da questo menu (**F.F.**) sono attivabili tutte le funzioni che consentono la manipolazione dei singoli files. La prima riga in alto sullo schermo visualizza (a sinistra) la versione di PCT e (a destra) l'eventuale nome di cui è dotato il dischetto selezionato in quel momento. In alto appare, come pro - memoria per l'utente, **il Path** attualmente selezionato per consentire di "orientarsi" tra i nomi dei files e, soprattutto, per sapere in qualsiasi momento su quale dischetto e su quale directory si sta operando. Se un particolare **wildcard** (punto interrogativo e/o asterisco) è stato selezionato, viene visualizzato anch'esso; altrimenti, in coda al path, viene visualizzato il classico ***.*** (asterisco, punto, asterisco) che indica la volontà di operare su tutti file presenti, senza alcuna distinzione.

ESC. In qualunque momento il tasto Esc serve per "uscire" dall'applicazione selezionata (magari involontariamente) in un certo momento. Se, poi, si preme Esc per errore, un opportuno **requester** (tipo *Are you sure?*)chiede conferma prima di proseguire. Non c'è che dire: PCT è a prova di distratti...

Il comando A (attribute).

Come è noto, in ambiente Ms - Dos, ad ogni file è possibile assegnare quattro attributi. **Read only**; se posto in **On** sarà possibile solo leggere tale file; se, al contrario, si è in **Off** sarà possibile, se necessario, modificarlo. E' ovvio che i programmi (professionali e non) di solito tengono conto di tale attributo e, di fatto, viene impedita la modifica del contenuto del file se questo attributo è su Off. **Hidden** (= nascosto); il nome del file, se l'attributo corrispondente è su **On**, non verrà visualizzato, ad esempio, con il comando **Dir**. Esempi di files nascosti sono rappresentati dai files di sistema e da altri files che, di solito, svolgono compiti particolari. I files di sistema, inoltre, hanno settato come **On** l'attributo **System**. La manipolazione degli attributi Hidde e System può essere pericolosa se compiuta con troppa disinvoltura. L'attributo **Archive**, se posto su **On**, permetterà, a successive manipolazioni del file (con, ad esempio, il comando Dos **Xcopy**) di effettuare selezioni dei files, dotati di tale attributo, in modo più spedito. Oltre ai quattro attributi, è possibile alterare la data e l'ora relativa al file selezionato.

Informazioni sui files. Ad ogni file, come è noto, è associato un gruppo di informazioni che lo caratterizza. PCT consente di visualizzarle in due modi: abbreviato ed esteso. Per default compaiono, per ciascun file, il **nome** (max. 8 caratteri), l'eventuale **estensione** (max. 3 caratt.), la **lunghezza** (size, misurata in bytes), gli **attributi** (vedi riquadro a parte), la **data** di memorizzazione. Nel **modo esteso** (attivabile con la pressione del tasto **F2**) oltre alle precedenti informazioni compare, per ciascun file, l'**ora** di memorizzazione, la traduzione degli attributi (eventualmente presenti) ed il numero di **cluster** del dischetto occupati dal file stesso (1 clu = 1024 byes).

Help

Premendo il tasto **H** viene visualizzato, in due schermate successive, un elenco (forse troppo stringato) delle funzioni disponibili in File Funcions. Inutile dire che tale elenco serve solo all'utente esperto, che ricorre alla funzione di Help solo nei casi in cui decide di usare comandi di raro uso.

```
PC Tools Deluxe                                          Vol Label=SANDRO
----------------------------------File Functions-----Scroll Lock OFF
Path=A:\*.*
    Name     Ext      Size #Clu   Date     Time  Attributes
    14GENN   ZIP    113093  111  2/13/91  5:31p Normal,Archive
    PRN2FILE COM      1386    2  6/01/90 22:00p Normal,Archive
    PROVA1            4104    5  2/12/91  4:27p Normal,Archive
 ----------------------------------------------------------------------
   3 files LISTed   =   118583 bytes.    3 files in sub-dir =   118583 bytes.
   0 files SELECTed =        0 bytes.   Available on volume =   156672 bytes.
-----------------------------------------------------------------------
   Copy Move cOmp Find Rename Delete Ver view/Edit Attrib Wordp Print List
Sort Help <ES=><+> F1=UNselect F2=alt dir lst F3=other menu Esc=exit PC Tools
  F8=directory LIST argument  F9=file SELECTion argument  F10=chg drive/path
```

Hard copy della schermata generale del menu File Functions

Nella schermata di PCT sono visualizzate tutte le parole chiave dei comandi disponibili. Per attivarli è sufficiente premere la lettera corrispondente ad ognuno di essi.

Copy

La funzione Copy ha un compito diverso da quello svolto dall'omonima funzione del menu Disk and Special Functions. Nel menu File Funcions, infatti, si possono effettuare **copie di files** singolarmente indicati premendo il tasto Return dopo essersi posizionati sul nome dei files desiderati; questi vengono numerati a mano a mano che si procede con la selezione e, in caso di errore, è possibile annullare la selezione di uno (o più files) premendo nuovamente return dopo essersi posizionati sui nomi. E' importante sottolineare che, ad ogni pressione di return, alcune informazioni presenti nella schermata vengono automaticamente aggiornate: si tratta del **numero dei files** (accumulati grazie alla selezione effettuata) e della loro **lunghezza** totale, espressa in bytes. Tali informazioni sono preziose per evitare tentativi di copia verso un supporto magnetico incapace ad ospitarli tutti.

Dopo aver selezionato i files bisogna premere il tasto **C** per richiamare la funzione di copia. Opportuni requester guidano la scelta del drive e della subdirectory in cui "scaricare" i files. Nel caso in cui esista già un file con lo stesso nome, una domanda sull'aggiornamento viene posta, per evitare danni irreparabili al vecchio file dotato di nome identico: **R** (=**re**place, sovrascrive **tutti** i vecchi file con quelli nuovi eventualmente dotati di nome identico). **W** (**w**rite = sostituisce **solo** il file indicato). **S** (evita la copia di **tutti** i files dotati di nome identico). **T** (evita solo la copia del file indicato)

Ver e Edit

Un file, come è noto, potrebbe presentare problemi nel caso, soprattutto, si sia utilizzato un supporto magnetico scadente oppure si sia "pasticciato" con manipolazioni pericolose sul file stesso o sui suoi attributi. Un comando di verifica, attivabile con il tasto **V**, compie un esame approfondito sul contenuto del file indicando, se è il caso, il luogo fisico del dischetto in cui si è verificato un inconveniente di qualsiasi tipo.

Find

Questa comoda opzione consente di rintracciare, all'interno di un qualsiasi file, la presenza di un gruppo di caratteri (fino a 32), da indicare digitandoli direttamente oppure mediante il codice esadecimale corrispondente. Tale opzione (da attivare con **F**) è comodissima, ad esempio, per effettuare personalizzazioni di schermate di programmi inglesi in cui si desidera sostituire i messaggi, ivi presenti, con le corrispondenti traduzioni in italiano

List

Stampa l'elenco dei files presenti nella directory selezionata.

La stampa risulta piuttosto ordinata e contiene in modo ben leggibile tutte le informazioni sui singoli files (nome, estensione, lunghezza, n. cluster, data, ora, attributi).

L'output è completato da una breve statistica.

Compare

Spesso, soprattutto lavorando con w/p, si ha a che fare con files che hanno lunghezza identica ma nome diverso (oppure hanno anche lo stesso nome, ma si trovano su dischi e/o directory diversi). In questi casi non si riesce a ricordare se i due files sono identici (in seguito, ad esempio, ad una copia effettuata per sicurezza) oppure se c'è una differenza, anche se trascurabile. Il comando di confronto (da attivare con il tasto **O**) funziona, ovviamente, solo se la lunghezza dei due files è identica. Una volta fatta partire la procedura, se i due files presentano una qualche differenza viene visualizzato il numero del byte ed il settore relativo in cui questa è stata notata. L'utente, quindi, può decidere in seguito se intervenire sul file per apportare le dovute modifiche, magari usando il comando Edit.

Wordp

Premendo il tasto **W** viene creato un file Ascii oppure, se è stato selezioanto un file prima della pressione di W, questo viene visualizzato come un file di testo in cui apportare eventuali modifiche. Il comando è di una comodità straordinaria per la creazione di files **Batch**, che possono essere editati, corretti, copiati e modificati con una facilità incredibile. Tentando di usare il comando per leggere files creati da altri w/p (come **Word Star**) si rischia di non ottenere la leggibilità del testo a causa della presenza dei caratteri speciali, tipici del w/p che ha generato il file

Move

Opera in modo analogo a **Copy**, solo che, dopo aver effettuato la copia sul disco / directory di destinazione, cancella i files indicati nel path originario. Comando utilissimo per gli utenti che, dotati di un solo hard disk e di un solo floppy, sono costrettui ad usare intensivamente un dischetto come "buffer". In caso contrario sarebbero costretti ad effettuare "a mano" la cancellazione dei files sul dischetto - buffer, allo scopo di ottenere spazio per ripetere operazioni di copia.

Rename e Delete

Per cambiar nome ai files occorre dapprima selezionarli (con il tasto Return) e poi premere il tasto **R**. Allo stesso modo si agisce per il comando **D**elete. In quest'ultimo caso, per sicurezza, viene chiesto se si desidera cancellare tutti i files indicati (senza alcuna conferma) oppure se si desidera cancellarli uno per uno annullando il comando, se è il caso, per uno o più files, benchè selezionati in precedenza. In caso di ripensamenti, come già detto, basta premere il tasto **Esc**.

Print

Dopo aver selezionato un file (ovviamente di testo) è possibile inviarlo su stampante così com'è (opzione **P**) oppure (opzione **W**) attribuendogli caratteristiche degne di un vero e proprio word processor: si può indicare il numero di linee per pagina, settare i margini destro e sinistro, imporre la numerazione, una pausa tra una pagina e la successiva ed altre opzioni utili.

Sort

Per meglio raccapezzarsi tra un elevato numero di files, è possibile visualizzare il loro nome in vari ordini: si impone l'ordine alfabetico dei nomi (con il tasto **F7**); delle loro estensioni (con **F8**); in base alla lunghezza (**F9**); alla data e ora (**F10**). E' addirittura possibile che l'ordine indicato (crescente o decrescente) venga memorizzato permanentemente nella directory del dischetto.

Edit

Deve essere usato solo dagli utenti esperti perchè è in grado di esplorare i singoli bytes di un file e di apportarvi eventuali variazioni. Le manipolazioni più sofisticate si effettuano facendo visualizzare blocchi di **16** righe costituite da **16** bytes ciascuna, visualizzati in formato esadecimale e Ascii. Il modo di operare con questa sosfisticata opzione ricorda il famoso **Zoom** del **C/64**.

I tasti speciali

Dal menu **File Functions** sono disponibili particolari funzioni, valide per numerosi comandi, attivabili premendo alcuni dei tasti funzione. **F2** è stato già esaminato in precedenza; con **F1** è possibile, con un colpo solo, **annullare** la selezione dei files effettuata, ad esempio, per errore oppure dopo aver usato un determinato comando che consente la selezione multipla dei files. **F8** consente di visualizzare soltanto i files aventi particolari caratteristiche (si possono inserire wildcards nel nome e/o nell'estensione); con **F9**, invece, si visualizzano ancòra tutti files, ma vengono automaticamente selezionati quelli aventi determinate caratteristiche (anche qui messe in evidenza ricorrendo a wildcards); quest'ultimo comando risulta comodissimo, ad esempio, quando si vogliono cancellare tutti files con estensione .BAK oppure copiare tutti i .DOC, eccetera. Premendo, in rapida successione, **F9** ed **F10**, vengono automaticamente selezionati **tutti** i files presenti nella videata.

Da notare che è sempre attiva, in tempo reale, una **statistica** relativa al numero dei files selezionati (manualmente o automaticamente) ed alla loro lunghezza. **F10**, se premuto da solo, consente di selezionare altri drive e/o directory.

SPECIAL FUNCIONS

Comandi già visti

La videata del menu Special Functions è praticamente identica a quella offerta da File Functions. La differenza consiste nel fatto che le varie operazioni, anzichè riferirsi ai singoli files, si riferiscono all'intero dischetto.
Ad esempio, con il comando Copy (tasto **C**) si possono duplicare interi dischetti. E' sufficiente indicare la lettera corrispondente al dischetto sorgente (**source**) e quella che si riferisce al dischetto destinazione (**target**). Il formato dei due dischi, ovviamente, deve essere identico (entrambi da 3.5 oppure 5.25) altrimenti la copia non viene effettuata; nel caso non si disponga di molta memoria, l'operazione viene effettuata in più riprese.
Allo stesso modo, il comando che effettua comparazioni (tasto **O**) verifica l'eguaglianza di due dischetti, necessaria se si hanno dubbi, ad esempio, sull'efficacia di una copia effettuata in precedenza.
Anche la ricerca (comando Find, tasto **F**) di un gruppo di caratteri viene effettuata sull'intera superficie del disco; nel caso in cui il gruppo viene rintracciato, PCT ne evidenzia il settore. Dal momentoi che la ricerca può anche durare molto tempo, la pressione di Esc può interrompere le operazioni.

Raname (tasto **R**) consente di attribuire un nuovo nome al disco; Verify (tasto **V**) effettua una verifica dell'intero disco; Edit (tasto **E**) consente di posizionare la testina di lettura / scrittura in una parte fisica qualunque del disco (e di alterarla, eventualmente, a volontà); il comando help (tasto **H**) svolge funzioni analoghe a quelle già viste. Infine **F3** ed **F10** consentono di selezionare i menu principali di PCT. Con Esc (potevate dubitarlo?) si abbandona l'ambiente PCT.

Map

Questo comando (attivabile con il tasto **M** e disponibile solo nel menu Special Functions) visualizza schematicamente una mappa della superficie del disco selezionato. Non ha altra funzione se non quella di dare un'idea della sua "saturazione" e dell'eventuale "disordine" verificatosi in seguito a numerose cancellazioni e successive scritture di files. Tuttavia visualizza anche informazioni importanti, come la presenza di files nascosti, i settori allocati e non, eventuale presenza di settori danneggiati, e così via. Mentre la mappa è visualizzata, si può premere il tasto **F** per controllare la zona del disco occupata da un file (da specificare in seguito): la sua ubicazione verrà rappresentata da tanti quadratini chiari quanti sono i cluster occupati. Non è che serva a molto, ma è di un certo effetto...

Directory maint

Premendo il tasto **D** si "entra" in un altro menu che consente la manipolazione di una o più **subdirectory**. Con **F1** se ne può cambiare il nome; con **F2** è possibile crearne una nuova; con **F3** se ne può cancellare una (a patto che sia **vuota**, altrimenti il comando non viene eseguito); con **F4** (**prune** and **graft**) si può spostare una subdirectory da una "ramificazione" ad un'altra; in pratica è possibile personalizzare l'**albero** delle directory presenti in un dischetto. L'operazioine, però non deve esser compiuta con leggerezza. E' infatti noto che alcuni programmi funzionano correttamente se sono in grado di individuare determinati files all'interno di un **path** (percorso di directory) prestabilito. Alterando tale percorso con il comando Prune and Graft, i programmi potrebbero non funzionare più. Proprio per queto motivo PCT richiede numerose conferme prima di portare a compimento l'operazione; la pressione di **F4** consente di selezionare il percorso desiderato.

Info

Vengono visualizzate, premendo il tasto **I**, numersose informazioni generali: numero dei drive presenti, sistema operativo attivo, "riconoscimento" della presenza del coprocessore matematico, porte seriali e parallele installate, memoriua Ram presente eccetera.

Park

Parcheggia le testine di lettura / scrittura dell'hard disk. L'operazione, come è noto, è consigliata prima di effettuare un trasporto del computer. Durante un viaggio, infatti, urti accidentali potrebbero danneggiare irrimediabilmente i delicati meccanismi del disco rigido. Ricordiamo ai distratti che un qualsiasi computer, il cui hard disk risultasse "parcheggiato", lo sblocca non appena viene data tensione all'allapparecchio.

Locate

Il comando (attivabile con **L**) consnete di verificare la presenza sul disco di un gruppo di files dotati di caratteristiche comuni, ovunque essi siano presenti. Se, ad esempio, vogliamo individuare tutti i files di tipo .COM memorizzati su un dischetto, il comando locate esaminerà (directory per directory e file per file) l'intero contenuto del disco. In caso positivo visualizzerà, indicando il path completo per ciascun file, le principali informazioni (nome, estensione, lunghezza, data ed ora) relative al file stesso.

Inizialize

E' il pericoloso (quanto provvidenziale) comando di **formattazione** di un dischetto, attivabile con la pressione del tasto **N**. PCT riconosce automaticamente il formato del drive e, nel caso siano possibili più tipi di formattazione, richiede di indicarla. Data la delicatezza dell'operazione, numerose conferme vengono poste prima di dare avvio all'operazione.

Undelete

Il miracoloso comando (attivabile premendo **U**) permette di recuperare un file, o una directory, accidentalmente cancellati. Naturalmente l'operazione ha successo se, dopo la cancellazione, il dischetto non è stato adoperato per memorizzare altri dati; in questo caso potrebbero essersi verificate sovrapposizioni che impedirebbero il recupero del file. Prmendo U compaiono solo i nomi dei files cancellati in cui, al posto del primo carattere, è presente un punto interrogativo. Per come è strutturato l'Ms - Dos, infatti, il primo carattere del nome contiene altre informazioni. Digitando il carattere, dunque, è possibile recuperare il file in modo automatico (tasto **F1**) oppure manuale (**F2**); quest'ultima opzione (da usare con cautela e solo se siete davvero esperti) consente, talvolta, di recuperare anche files "irrecuperabili".

di Alessandro de Simone

IL COMPUTER AUGURA BUON COMPLEANNO

Ricordare a memoria le date importanti (e non) può, a volte, essere problematico. Un po' di "fosforo" informatico non guasta...

Chi, come un professionista, ha a che fare quotidianamente con appuntamenti e impegni di vario tipo, non può permettersi il lusso di evitare l'uso di un'agenda in cui segnare tutto ciò che è importante ricordare.

Gli scippatori seri, infatti, sanno benissimo che l'agenda degli appuntamenti è di vitale importanza per un avvocato e, dopo avergliela sottratta (spesso con sotterfugi poco educati) richiedono, al professionista, un "riscatto" rilevante per rientrarne in possesso. Altri, invece, ricorrono a nodi al fazzoletto, oppure a segretarie assunte specificamente per svolgere uno scopo determinato (cioè per segnare tutti gli appuntamenti; che avevate capito?...). Chi, come noi comuni mortali, ha poche (ma essenziali) date da ricordare, si limita a segnarle sul calendario che, in seguito, abbandona spudoratamente in un cassetto.

A ricordarci del compleanno mancato provvederà la tradizionale, quanto gelida telefonata (successiva e non precedente, badate bene, al genetliaco dimenticato) della persona interessata che non mancherà di farci notare l'alto livello di maleducazione raggiunto da inqualificabili (ma altrettanto identificabili) individui.

In effetti, noi smanettoni usiamo il computer molto più spesso di un'agenda, illudendoci, magari, di avere ancora una buona memoria per ricordare altre cose terrene. Inevitabile, a questo punto, una domanda: perchè non **realizzare una procedura automatica che, non appena accendiamo il computer, sia in grado di rammentarci le prossime scadenze**? Detto, fatto.

Hard Disk ed Autoboot

La procedura che stiamo per descrivere deve esser considerata solo la **base** per lo sviluppo di un programma, e merita di essere migliorata. Ci limiteremo, pertanto, ad accennare alle caratteristiche che una procedura del genere deve possedere per essere automatizzabile. Anzitutto c'è da dire che possono essere usati computer dotati di "automati-smi" intrinseci. Supponiamo, infatti, di aver realizzato un programma di nome **Agenda** che, **solo se** caricato, consenta di effettuare la ricerca di date "importanti". I casi sono due: dimentichiamo di attivare frequentemente "Agenda" (ed un oblio di tal genere può esser fatale); oppure, effettivamente, ricordiamo di caricare "Agenda" tutti i giorni, non appena ci mettiamo al computer; in questo caso, però, risulterebbe molto più semplice gestire una tradizionale agenda che richieda l'uso di carta e penna!

No: **la procedura deve partire automaticamente** ogni volta che accendiamo il computer, anche se (o meglio, soprattutto se) dimentichiamo l'approssimarsi di date importanti.

Tutti i moderni computer, fortunatamente, dispongono di una procedura automatica che viene attivata non appena si accende l'apparecchio.

Sui sistemi **Ms - Dos** il File Batch in questione si chiama **Autoexec.bat**; su Amiga, invece, il nome del file è **StartUp-Sequence**.

In entrambi i casi, ogni riga del file (personalizzabile, quest'ultimo, dall'utente) rappresenta un **comando** del Dos, o un **programma** da lanciare in tale ambiente. In effetti non è strettamente ne-

cessario che sia disponibile, all'accensione, il file di Auto - Boot; tuttavia c'è quasi sempre l'esigenza del lancio di un certo numero di files (se non altro, quello relativo alla configurazione della tastiera) prima di adoperare il computer; tale necessità impone, di fatto, la presenza del file di Auto - Boot.

Un altro vantaggio dei moderni computer è rappresentato dal fatto che montano (quasi) tutti, al loro interno, un **orologio - datario** dotato di alimentazione autonoma, in grado di aggiornare la data e l'ora, anche nel caso in cui il computer sia spento.

Se, quindi, avete a disposizione un computer dotato di **hard disk** ed orologio interno alimentato da una batteria - tampone, la procedura che stiamo per descrivere sarà **totalmente automatica**.

Se non disponete di hard disk, dovrete prendere l'abitudine di usare sempre, per il "lancio" del sistema operativo, solo il dischetto che contiene il file di Auto - Boot realizzato nel modo che spiegheremo.

Se, poi, non disponete di orologio interno, sarete costretti a digitare, tutte le volte che "lancerete" il sistema operativo, la data "corrente".

Se, infine, non disponete di computer che richiede un file di Auto - Boot... arrangiatevi con un'agenda cartacea!

Il programma

Prima di procedere, è opportuno precisare che il funzionamneto della procedura richiede alcuni accorgimenti.

Anzitutto c'è bisogno che sia accessibile, sul dischetto di lancio (nel nostro caso, per un sistema Ms - Dos, il disco rigido denominato **C:**) una **subdirectory** dal nome **Data** che avrà il triplice scopo di: creare una "nicchia" riservata, atta ad ospitare soltanto i dati relativi alle date da ricordare (soprattutto per non fare confusione con i files di altri programmi); rintracciare facilmente i files di pro-memoria relativi a tali date; evitare l'occupazione di spazio, su video, durante l'eventuale visualizzazione della Root (directory principale) durante le normali operazioni di gestione del disco.

All'interno della directory Date saranno presenti: tutti i files Ascii relativi alle singole date da ricordare; il programma **Data.exe** compilato (realizzato con l'apposita opzione del linguaggio QuickBasic o altri compilatori) in grado, cioè, di esser lanciato da ambiente Dos senza la presenza del linguaggio stesso; il file dal nome **Data.dir** che, come dice il suffisso, rappresenta la *redirezione* del comando **Dir** sotto forma di file Ascii (vedi riquadro specifico in queste pagine).

Per comodità vi consigliamo di memorizzare, nella stessa dir Data, anche il programma sorgente (Data.Bas), allo scopo di evitare faticose ricerche dello stesso nel caso si vogliano apportare modifiche di vario tipo.

Un esempio del contenuto di Data.dir e, indirettamente, anche di una possibile directory Data, è pubblicato nella figura di queste pagine.

La redirezione dei files

E' probabile che non tutti i neo-utenti siano informati su un mini-trucchetto realizzabile con i più noti comandi del Dos.

Qualunque sia il Sistema Operativo (O.S.) usato, esiste un comando (DIR) che consente di visualizzare, sullo schermo, il contenuto di una certa directory.

Se, invece di un semplice comando Dir, digitiamo...

Dir > prn

...la directory verrà "inviata" alla stampante (se, ovviamente, presente e... accesa) anzichè sullo schermo. Allo stesso modo, con...

Dir > NomeFile

...ritroveremo, sulla directory attuale, un file Ascii, di nome NomeFile, contenente l'elenco dei files contenuti nella Directory. Se, ovviamente, digitiamo...

Dir > c:\data\data.dir

...ritroveremo il file Ascii di nome Data.dir, dopo l'esecuzione del comando, all'interno della directory, presente nel drive C:, di nome Data. Inutile dire che la directory Data deve essere già presente, altrimenti il comando non avrà effetto.

Se, ora, impartiamo il comando...

Type Nomefile

...oppure...

Type c:\data\data.dir

...(a seconda di ciò che abbiamo prima digitato), constateremo che il file di nome Data.dir effettivamente esiste e contiene ciò che ci aspettavamo.

La redirezione, e concludiamo, si "applica" anche ad altri comandi. Ad esempio, se a questo punto delle operazioni digitate...

Type c:\data\data.dir > prn

...il file Ascii verrà riversato sulla stampante.

```
Il Volume nell'unità C non ha etichetta
 Indirizzario di C:\DATA

.              <DIR>       1/02/91   22:27
..             <DIR>       1/02/91   22:27
910305              116    3/02/91   18:07
910208               73    3/02/91   18:11
DATA      DIR         0    3/02/91   16:17
DATA      BAS      1372    3/02/91   18:00
910211               57    3/02/91   18:12
910213               27    3/02/91   18:12
DATA      EXE     35484    3/02/91   18:01
910210               62    3/02/91   18:13
911128               47    3/02/91   18:14
920110               42    3/02/91   18:15
       12 file
   245760 byte disponibili
```

Un possibile contenuto del file Data.Dir

Il trucco

La notevole semplicità del programma pubblicato richiede, da parte del lettore, un minimo di accorgimenti.

Anzitutto, come già precisato, questi deve creare, con...

md c:Data

...la directory di nome Data destinata ad ospitare i vari files che serviranno per rendere automatica la procedura. In seguito, caricato il linguaggio **QuickBasic Microsoft**, digitare e registrare, nella stessa directory, sia il programma sorgente (dal fantasioso nome **Data.bas**) sia il programma eseguibile, comprendente

tutte le librerie per "sganciarlo" dal linguaggio stesso (il nome del file sarà l'altrettanto fantasioso **Data.Exe**).

A questo punto, per provare il corretto funzionamento del programma, "entrate" nella dir Data e registrate, magari servendovi di una qualsiasi versione di **PcTools** (o, se ne siete sprovvisti, del programma **Edlin** presente tra i file-programmi Ms - Dos) alcuni files di testo il cui nome sia formato da **sei** caratteri.

I primi due di questi rappresenteranno l'**anno** (esempio: **91**) della data da ricordare; i successivi due indicheranno il **mese** (**11** = novembre; 04 = aprile, e così via); gli ultimi due indicheranno il **giorno** della data fatidica (anche in questo caso, se la data è inferiore a 10, digiterete uno zero inziale; esempio: 02, 08, 31, eccetera). In questo modo, ad esempio, il file di nome **910730** conterrà informazioni relative alla data **30 luglio 1991**.

Tale accorgimento, in effetti, non è indispensabile ed è possibile usare altre codificazioni per ricordare la data. Il programma pubblicato, ovviamente, terrà conto della codificazione accennata.

Che cosa digiterete **nel** file "aperto" di nome 911128? Ovvio: il messaggio che consentirà di ricordare il motivo che ci ha spinto a memorizzare la data stessa; in aggiunta, perchè no, potrete memorizzare altre notizie; relative, magari, ad altri avvenimenti vicini alla data in questione.

Inutile dire che, non appena vi verrà in mente una data da ricordare, dovrete accendere il computer e memorizzare il pro - memoria relativo all'avvenimento stesso, nel modo appena descritto, dotando il file di un nome tale da rispettare la codifica necessaria alla successiva interpretazione da parte del programma.

Non dovrebbero esserci dubbi, ora, nell'interpretare il significato dei files contenuti nella directory Data. Inutile dire che sarà anche necessario modificare il file di auto - boot. Nel nostro caso (sistema Ms - Dos) sarà sufficiente che l'ultima parte del file **Autoexec.bat** contenga i seguenti comandi...

```
cd data
dir > data.dir
data.exe
```

Il **primo** consente di "entrare" nella directory Data; il **secondo** redirige la stessa sub-directory nel file Data.dir (che, in questo modo, verrà automaticamente aggiornato ad ogni reset o accensione del computer); il **terzo** comando, infine, lancia il programma Data.exe, incaricato di verificare l'approssimarsi di date importanti. Chi ha l'**Amiga** dovrà utilizzare un Basic compilabile, altrimenti la procedura che carica dapprima il linguaggio **AmigaBasic** e, successivamente, lanci Data.Bas, potrebbe rivelarsi troppo lunga ed inefficiente, benchè perfettamente funzionante.

Come funziona

Il listato di queste pagine, scritto in **QuickBasic Microsoft**, è talmente breve e semplice da comprendere che anche chi possiede un Amiga (o è abituato ad usare altri linguaggi) può effettuare gli adattamenti del caso.

Nelle prime righe si rendono "globali" alcune variabili, in modo da renderle accessibili anche da subroutines o funzioni che il lettore volesse aggiungere per migliorare il programma.

In due vettori, il primo numerico (**Mese**), il secondo stringa (**Mese$**) vengono allocati, rispettivamente, il numero di giorni trascorsi dall'inizio dell'anno ed il nome del mese stesso.

Ad esempio, Mese(2) contiene il valore 31; **Mese(3)**, invece, contiene **59**; questo numero indica che, quando è selezionata la stringa **Mese$(3)** (cioè **Marzo**) sono trascorsi, dall'inizio dell'anno, 59 giorni (**31** di gennaio + **28** di febbraio). Ciò implica, nella fase successiva di decodifica del nome del file, una certa agevolazione nel determinare il numero di giorni che mancano alla data da ricordare.

E', infatti, quasi inutile ricordare una data il giorno stesso della sua ricorrenza: sia che questa si riferisca ad un compleanno o alla scadenza di una tassa, il lasso di tempo per provvedere alla bisogna (acquisto del regalo, raccolta di eventuale documentazione) potrebbe rivelarsi insufficiente. Inoltre non dobbiamo dimenticare che, benchè smanettoni incalliti, potrebbe capitarci di non usare il computer per qualche giorno. Meglio, quindi, avere la possibilità di un congruo preavviso, del resto modificabile a volontà alterando la riga...

```
If (Totale - y) < 6 Then
```

...rintracciabile nella parte finale del listato pubblicato.

```
Oggi, 28 novembre, è il
compleanno di Fabrizio.
Tempo fa ha manifestato il
desiderio di possedere
tutti i CD di John Lennon.
Telefonare per verificare
quali mancano alla sua
collezione e provvedere
per il regalo.
Ricordarsi che tra breve
i negozi saranno aperti
anche di domenica.
```

Esempio di "contenuto" del file 911128

Il comando **Open** si incarica di leggere il file Data.Dir, pluricitato nel corso dell'articolo; le istruzioni successive, invece, agiscono in modo da eliminare, dall'elaborazione, tutte le righe ed i files che non hanno nulla a che fare con le date da ricordare. Il file Data.Dir, infatti, contiene TUTTI i caratteri Ascii visualizzabili con il comando Dir, compresi il numero dei files contenuti ed il numero di bytes ancora disponibili sul drive. Opportuni "filtri", che agiscono grazie all'uso intensivo di **Val**, **Left$**, **Right$**, **Mid$**, consentono di "estrarre", dal file Data.Dir, solo i nomi dei files che interessano.

In base alla codifica seguita, il programma individua, seleziona e visualizza i nomi-data dei files la cui ricorrenza scade entro cinque giorni.

In altre parole, non appena accenderemo il computer possono verificarsi due casi: una (o più) date sono considerate vicine; oppure nessuna ricorrenza avrà luogo entro i prossimi cinque giorni.

Nel primo caso verrà visualizzata la data odierna, ed il file batch (Autoexec.bat) continuerà per la sua strada. Nel secondo caso, invece, non solo verranno emesse tante segnalazioni acustiche (notare la presenza del comando **Beep**) quante sono le date imminenti; ma il programma attenderà la pressione di un tasto prima di proseguire.

Se, infatti, dopo aver acceso il computer ci distraessimo, potremmo non notare l'eventuale visualizzazione dei messaggi; i successivi comandi contenuti nel file di auto - boot, inoltre, potrebbero cancellare lo schermo, impedendoci di prender nota delle date evidenziate.

```
DIM SHARED x$, x1, x2, x3, x   : REM Programma in QuickBasic Microsoft
DIM SHARED Mese(12), Mese$(12): REM per la visualizzazione automatica delle date
DECLARE SUB esamina (x$, x1, x2, x3)
FOR i = 1 TO 12: READ Mese(i): NEXT
FOR i = 1 TO 12: READ Mese$(i): NEXT
DATA 0,31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334
DATA Gennaio, Febbraio, Marzo, Aprile, Maggio, Giugno, Luglio, Agosto
DATA Settembre, Ottobre, Novembre, Dicembre
CLS
y1 = VAL(RIGHT$(DATE$, 2)): PRINT "Anno: 19"; RIGHT$(STR$(y1), 2); : REM anno attuale
y2 = VAL(LEFT$(DATE$, 2)): PRINT "   Mese: "; Mese$(y2); : REM mese attuale
y3 = VAL(MID$(DATE$, 4, 2)): PRINT "    Giorno: "; y3: REM giorno attuale
PRINT : y = Mese(y2) + y3: REM N. giorni trascorsi da inizio anno
Esistono = 0
OPEN "c:\data\data.dir" FOR INPUT AS #1
DO WHILE NOT EOF(1): REM Legge fino alla fine del file
INPUT #1, x$: x = LEN(x$): Esiste = 0:
IF x > 0 AND VAL(LEFT$(x$, 6)) > 910000 THEN Esiste = 1: Esistono = 1
   IF Esiste = 1 THEN
       CALL esamina(x$, x1, x2, x3)
           IF x2 < 13 AND x2 > 0 THEN
               totale = Mese(x2) + x3 + (x1 - y1) * 365
           END IF
           IF (totale - y) < 6 THEN
               BEEP: PRINT "Mancano "; totale - y;
               PRINT "giorni al"; x3; Mese$(x2);
               PRINT " 19"; RIGHT$(STR$(x1), 2), "(cfr. file "; LEFT$(x$, 9); " )"
           END IF
   END IF
LOOP
PRINT
IF Esistono <> 0 THEN
   PRINT "Premi un tasto per continuare"
   DO WHILE a$ = "": a$ = INKEY$: LOOP
END IF

SUB esamina (x$, x1, x2, x3)
x1 = VAL(LEFT$(x$, 2)): REM anno da ricordare
x2 = VAL(MID$(x$, 3, 2)): REM mese da ricordare
x3 = VAL(MID$(x$, 5, 2)): REM giorno da ricordare
END SUB
```

Migliorie

La procedura, volutamente semplice, consente, però, di commettere involontariamente alcuni errori. Ci riferiamo, principalmente, alla **digitazione del nome del file** che, se non presenta problemi per l'anno, può far sorgere confusione sulla corretta successione di mese e giorno. Non sarà un problema scrivere, a parte, un programmino che, oltre alla memorizzazione del messaggio, consenta di assegnare il nome del file senza commettere errori. Si noti che il programma, per come è scritto, sfrutta solo i primi sei caratteri (contro gli **otto** + **tre** del suffisso) disponibili in Ms - Dos; le elaborazioni, inoltre, agiscono solo su tali sei caratteri. Vi suggeriamo di introdurre una modifica in grado di assegnare un preavviso diverso (numero di giorni di anticipo o tipo di ricorrenza da evidenziare) a seconda del codice (da "inventare" a parte) assegnato agli altri due caratteri al suo suffisso. Per rendere più comoda la procedura si potrebbe introdurre un'opzione che, impartendo un semplice comando **Type**, consenta la visualizzazione dei files individuati dal programma; l'introduzione di una subroutine di **Sort** (ordinamento), inoltre, non guasterebbe di certo. Ricordiamo che l'aggiornamento delle date si può realizzare in modo semplicissimo: dopo aver provveduto, ad esempio, all'acquisto del regalo (caso del compleanno che càpita in data 7 maggio 1991) basterà cambiare il nome del file ad esso relativo (ciè **91**0507) in **92**0507 mediante il comando **Rename**. In questo modo, non solo verrete avvisati per tempo anche l'anno prossimo, ma eviterete che, dal giorno 2 maggio 1991 in poi, venga emessa la segnalazione di "prossima scadenza" tutte le volte che accenderete il computer.

Una sfida al mese

LA DIVISIONE INFINITA

Un listato talmente breve, che più breve non si può, costituisce il "nucleo" della sfida proposta ai lettori questo mese

Nel 1983 pubblicai un programma che simulava, su computer, il "ragionamento" compiuto da ognuno di noi quando svolgiamo una divisione.

Il programma (qui presente in due versioni, una più breve dell'altra) consente di ottenere il **quoziente** di due valori il cui numero di cifre è infinito (primo micro programma); nel caso del secondo listato, invece, il numero di cifre ottenibile varia tra 4 e 256 (caso del **C/64**) oppure raggiunge un valore compatibile con le potenzialità offerte dal linguaggio adoperato; il **Quick Basic** della Microsoft, ad esempio, accetta stringhe formate da diverse migliaia di caratteri. Il secondo programma, infatti, crea una stringa che, a mano a mano che l'elaborazione procede, viene allungata sempre più, di un carattere alla volta.

Inoltre viene effettuata una ricerca del **periodo** eventualmente presente, purchè di lunghezza non superiore alla metà della lunghezza massima consentita per la memorizzazione delle stringhe.

Come funziona

Facciamo notare che il programma, scritto in un **Basic** quanto più universale possibile, può girare su un qualsiasi elaboratore; per il C/64 l'unico cambiamento richiesto è quello relativo alla modifica delle istruzioni che sovrintendono alla individuazione della pressione di un tasto (come indicato, appunto, nel secondo listato).

Il principio da cui parte l'idea è meglio comprensibile se diamo uno sguardo al programma più breve. In pratica il computer viene costretto ad eseguire una divisione proprio come la faremmo noi, con carta e penna. Nelle prime due righe (**100** e **110**) vengono richiesti il **dividendo** ed il **divisore**. Nella riga **120** viene calcolata, in tutta fretta, la parte intera del quoziente, che può anche essere uno zero. La riga **130** si limita a visualizzare la virgola, dopo la parte intera, per differenziare quest'ultima dalla parte decimale.

Con la **140** inizia il lavoro vero e proprio: il **resto**, dal momento che ci troviamo "dopo" la virgola, deve esser moltiplicato per 10 prima di far proseguire il calcolo; in pratica l'operazione corrisponde al *"...e aggiungo 0..."* che siamo soliti fare, calcolando manualmente, scrivendo **0** a destra, nella colonna dei resti.

La riga **150** provvede ad aggiungere uno zero al quoziente nel caso in cui, nonostante sia stato aggiunto uno zero alla colonna dei resti, il resto stesso non risulti maggiore o eguale al divisore.

La coppia di righe **160** e **170** calcolano la cifra successiva.

Il periodo

Il secondo programma, basato sulla struttura del primo, consente la ricerca dell'eventuale **periodo**. Per quanto riguarda quest'ultimo, si è pensato di accumulare, nella stringa **P$**, tutte le cifre che vengono via via calcolate (ecco spiegato, tra l'altro, il limite del numero di caratteri); in seguito, mediante cicli For... Next, vengono comparate, fra loro, **sottostringhe** della stringa A$, allo scopo di rintracciare il periodo.

Non dobbiamo dimenticare, infatti, che in alcuni casi la parte periodica può "iniziare" a distanza di qualche cifra dopo la virgola.

E' ovvio che, servendosi di più stringhe, si può effettuare la ricerca del periodo su un numero di cifre più elevato, oppure effettuare operazioni tra valori con centinaia di cifre decimali.

Come gira

Dopo in Run, viene posta la domanda "Stampa continua?". Rispondendo **S** la visualizzazione delle cifre decimali continua finchè non si preme un tasto; rispondendo **N**, invece, viene richiesto il numero di cifre decimali desiderate dopo la virgola. A questo punto bisogna digitare il dividendo e, subito dopo, il divisore.

Nel primo caso, ovviamente, l'elaborazione continua all'infinito. Nel secondo, invece, dopo aver visualizzato il numero di cifre richieste dopo la comparsa della virgola, si assisterà ad una pausa, più o meno lunga (dipendente dal **computer**, dal **linguaggio** usato e dal **numero** di cifre richiesto) durante il quale il program-

```
100 INPUT "Dividendo"; D0
110 INPUT "Divisore"; D1
120 DX = INT(D0 / D1)
130 Q$ = STR$(DX) + ",": PRINT Q$;
140 RX = (D0 - DX * D1) * 10
150 IF RX < D1 THEN PRINT "0"; : RX = RX * 10: GOTO 150
160 Q = INT(RX / D1): PRINT RIGHT$(STR$(Q), 1);
170 RX = (RX - Q * D1) * 10: GOTO 150
```

Il micro listato Basic di partenza

Come partecipare

Siamo ora costretti a ripeterci, per la gioia dei nuovi lettori. Per partecipare alla sfida è necessario possedere un computer abbastanza veloce (in altre parole: **Ms - Dos** oppure **Amiga**) ed adoperare un linguaggio che non presenti problemi durante le prove generali; e lo diciamo nel vostro interesse: un computer AT compatibile, dotato di Quick Basic, a volte determina il periodo in un tempo piuttosto lungo; figuriamoci un C/64! Tuttavia, se il programma **non** viene scritto in l.m. e non si fa ricorso a **Peek** e **Poke**, accetteremo anche i lavori degli irriducibili 64isti.

Vengono ovviamente acccettati programmi nel linguaggio C (**Turbo C** oppure **Quick C**, per ciò che riguarda **Ms - Dos**; un qualsiasi **C** per Amiga) e Pascal (**Turbo Pascal** oppure **Quick Pascal**; solo ambiente **Ms - Dos**).

Dobbiamo proprio ripetere, per l'ennesima volta, le ultime raccomandazioni? Come volete: le maggiori probabilità di vincere la sfida le avranno coloro che scriveranno programmi **brevissimi**. Inoltre potranno partecipare alla sfida **esclusivamente** coloro che concorderanno, per telefono (**02/57.60.63.10**, solo il giovedì pomeriggio), le modalità di invio del materiale. Il migliore lavoro varrà compensato con una cifra che potrà raggiungere anche le 200 mila lire (stabilita ad insindacabile giudizio della Redazione).

```
100 REM Divisione infinita - Versione Gw - Basic AmigaBasic
110 REM (per il C/64 modificare opportunamente
120 REM  l'istruzione B$ = INKEY$ righe 250, 386
130 REM  vedi righe 245 - 385)
240 PRINT "Stampa continua ? (s/n)"
245 REM GET B$ ----- per C/64
250 b$ = INKEY$
255 IF b$ = "" THEN GOTO 245
260 co = 1: IF b$ = "s" THEN co = 0: GOTO 290
270 PRINT "Quante cifre decimali (4 - 254)": INPUT cd: cd = cd + 1
280 IF cd > 255 OR cd < 4 THEN PRINT "errore": GOTO 270
290 e1$ = "periodo = ": e2$ = "parte decimale = "
300 e3$ = "dividendo =": e4$ = "divisore ="
310 PRINT e3$; : INPUT d0
320 PRINT e4$; : INPUT d1: PRINT
335 PRINT e3$; d0: PRINT e4$; d1: PRINT
340 dx = INT(d0 / d1)
355 q$ = STR$(dx) + ",": PRINT "quoziente = "; q$;
360 rx = (d0 - dx * d1) * 10
370 i = i + 1: IF i = cd AND co = 1 THEN GOTO 460
385 REM GET B$ --- PER C/64
386 b$ = INKEY$
387 IF LEN(b$) THEN PRINT : PRINT "...eccetera.....": GOTO 530
390 IF rx = 0 AND co = 1 THEN a$ = a$ + "0"
400 IF rx = 0 THEN GOTO 520
410 IF rx < d1 AND co = 1 THEN a$ = a$ + "0"
425 IF rx < d1 THEN PRINT "0"; : rx = rx * 10: GOTO 370
430 q = INT(rx / d1): b$ = RIGHT$(STR$(q), 1): PRINT b$;
440 IF co = 1 THEN a$ = a$ + b$
450 rx = (rx - q * d1) * 10: GOTO 370
460 l = LEN(a$)
470 FOR j = 0 TO l / 2: b$ = RIGHT$(a$, l - j)
480 FOR i = 1 TO l / 2: x$ = MID$(b$, 1, i): y$ = MID$(b$, i + 1, i)
490 IF x$ = y$ AND i > 1 THEN PRINT : PRINT e1$
500 IF x$ = y$ AND i > 1 THEN PRINT k$: PRINT x$: GOTO 530
510 NEXT i: k$ = LEFT$(a$, j + 1): NEXT j: GOTO 530
520 PRINT : PRINT : PRINT e2$: PRINT a$
530 PRINT
```

ma effettuerà le ricerche necessarie per l'individuazione dell'eventuale parte periodica. Alla fine, su due righe successive, verranno visualizzate la parte non periodica e quella periodica, rispettivamente.

La sfida

Come ogni programma che si rispetti, anche questo presenta casi particolari che meritano una modifica dell'algoritmo sviluppato.

Ad esempio, chiedendo di effettuare la divisione **123** / **4567**, viene, sì, visualizzato correttamente il risultato "generale", ma al momento della stampa della parte non periodica appare, sullo schermo, il valore **0269323407** mentre, come "periodo", viene indicato il valore **05**. Uno sguardo al risultato generale ci fa capire perchè il programma è stato ingannato: ad un certo punto, e proprio subito dopo il valore 0269323407, vi sono le cifre 0505... che il computer, erroneamente, considera come una coppia di periodo "05".

La sfida, quindi, consiste nella modifica del programma affinchè vengano riconosciute situazioni anomale ed interpretate correttamente.

Chi non si accontènta, inoltre, può scrivere un programma che, ad esempio, dopo aver memorizzato (su dischetto) due stringhe ottenute, in successione, effettuando due divisioni con il programma pubblicato, ne calcoli il prodotto, cifra dopo cifra, come faremmo noi con carta e penna. Via libera, come intuitivo, anche a coloro che vogliano scrivere algoritmi per determinare radici quadrate (o cubiche), logaritmi e così via.

di Valentino Spataro

UN ATLANTE PER AMIGA

Un programma per creare cartine geografiche terrestri, lunari e perfino mappe stellari. E un listato commentato, tutto da studiare

Se volete... computerizzare l'Italia, questo è il programma che fa per voi; ma anche se siete astrofili non ve lo dovete perdere (come vedrete oltre). Infatti il programma permette di creare cartine geografiche sullo schermo del monitor e richiede soltanto poche operazioni: 1) copiare il programma, 2) introdurre i dati delle località (denominazione, latitudine e longitudine), 3) visualizzare il tipo di cartina desiderato (tra 4 tipi diversi).

Il programma e i risultati

Descriveremo, ora, le singole opzioni del programma per meglio spiegare in che modo è possibile produrre cartine geografiche in scala e mappe stellari. Per i programmatori "puri", il listato, ultracommentato, sarà certamente fonte gradita di **trucchetti** di vario tipo. Grazie alla sua semplicità, per quanto riguarda la gestione dei dati (vedi le sezioni **introduzione**, **modifica**, **elenco**, **ricerca**, **load** e **save**) è una chicca per chi ha sempre programmato sul C/64 ed ha ancora poca esperienza con un interprete Basic evoluto.

Si consiglia quindi vivamente lo studio del listato, sia a tavolino, sia dopo aver fatto girare il programma.

E' importante sottolineare che il programma è stato scritto settando **Preferences** sulla modalità di **80 colonne**; se l'impostazione è, invece, regolata su **60 colonne**, possono verificarsi **disallineamenti** in fase di visualizzazione. Si consiglia, pertanto (magari dopo aver trascritto il programma nella modalità che desiderate) di dare il Run solo dopo aver settato Preferences su 80 colonne.

New, Directory, Quit

Si premette che, alla partenza del programma e durante il suo uso, vengono di volta in volta attivate le sole opzioni dei menu da utilizzare per sapere in ogni istante quello che (non) si può fare.

Le quattro opzioni del menu **project** (selezionabili premendo il tasto destro del mouse dopo aver posizionato la sua freccia sulla parola Project) permettono rispettivamente di: (**New**) azzerare le variabili e far ripartire il programma dall'inizio; (**Directory**) elencare il contenuto di un dischetto; (**Quit**) ristabilire il menu standard dell'editor del Basic e abbandonare il programma. Delle opzioni **Load** e **Save** parleremo dopo.

Default, Load e Introduzione / Modifica

Una volta lanciato il programma, le prime opzioni da usare sono quelle visualizzabili nel menu **Cartina**. La prima (**Default**) permette di variare alcuni parametri standard: il **nome** del pianeta (di cui volete rappresentare la cartina geografica) ed il suo **raggio**.

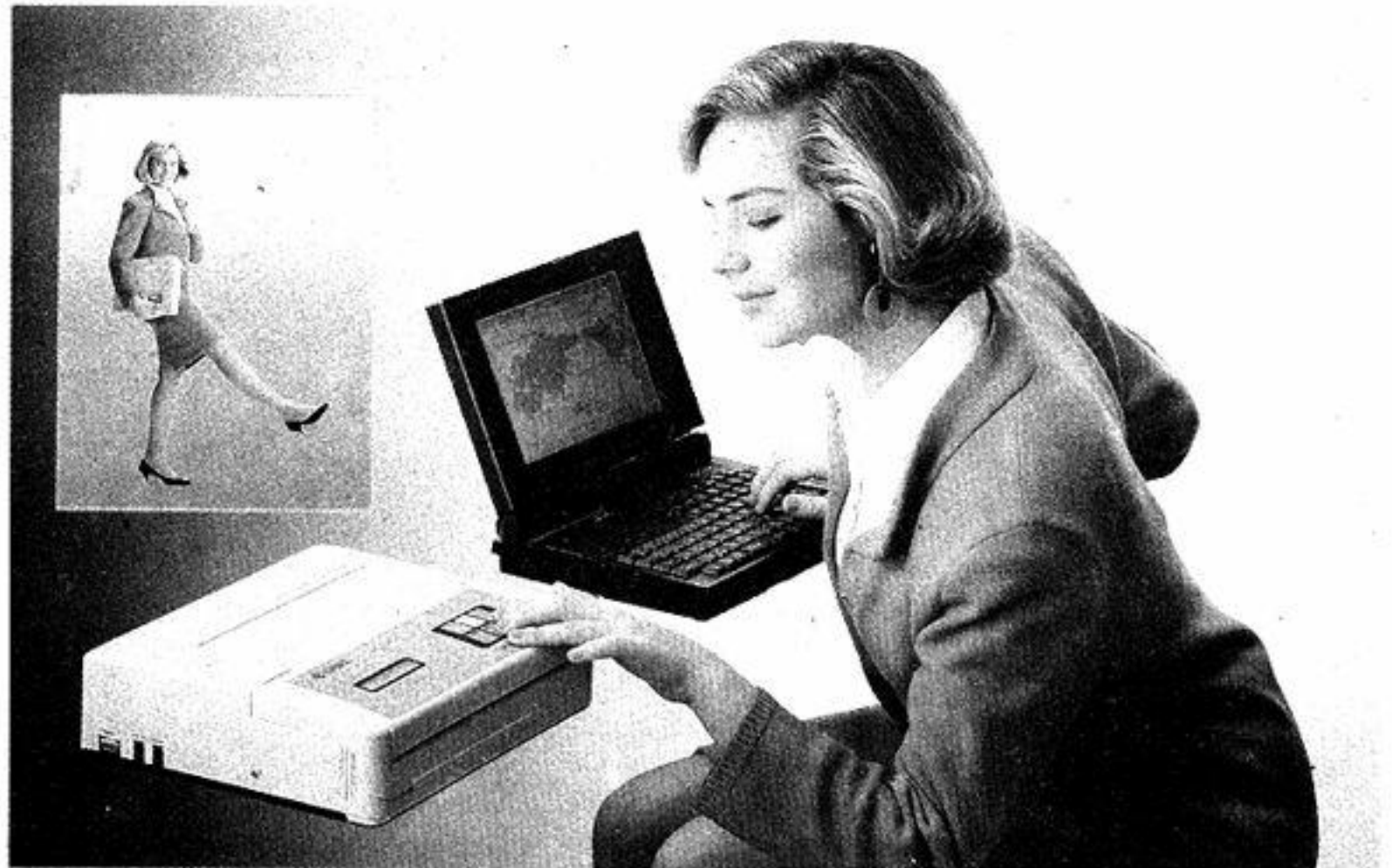
Il programma, infatti, permette di mantenere le proporzioni, in una scala abbastanza precisa, di vari punti di cui si forniranno **latitudine** e **longitudine**. In pratica, potrete usare il listato per tracciare punti significativi di un qualsiasi pianeta, limitandovi ad indicarne il solo raggio.

Nel caso vogliate tracciare cartine geografiche della **Terra**, i dati risultano già inseriti e non avete bisogno di selezionare questa opzione. Se, però, volete visualizzare una cartina con i crateri lunari dovrete inserire: **Luna** e **1738** che è il suo

raggio espresso in km. Nel caso di una **mappa stellare** (per visualizzare, ad esempio, le costellazioni) non avete bisogno di cambiare raggio, in quanto in ogni caso viene sfruttato al massimo lo schermo; solo l'indicazione della scala (calcolata automaticamente dal programma) sarà, ovviamente, errata, pur restando le proporzioni esatte.

Dopo aver impostato i dati di default, bisogna mettere in memoria i dati delle località. Dovrete cioè (solo la prima volta che userete il programma) inserire i dati relativi ad ogni località (sia esso lago, città, monte, etc.) che sono: **denominazione**, **latitudine** e **longitudine**, dati espressi in gradi. Il programma provvede, poi, ad inserire, nelle matrici **lat # ()** e **lon # ()** i valori in radianti e a calcolare le coordinate massime e minime (allo scopo di determinare il centro dello schermo e la scala migliore).

Scegliendo **Introduzione - Modifica** (sempre dal menu Cartina) verrà chiesto il codice su cui operare: inserendo quello consigliato potrete aggiungere dati nuovi, altrimenti modificare un codice già esistente. Non immettendo alcun codice tornerete al menu principale.

Con **Load** (menu Project) potrete scegliere di tenere in memoria i dati già presenti o cancellarli. In memoria non

possono esservi più di **500 dati**, ma variando il valore della variabile **Max**, posta all'inizio del programma, potete aumentare la quantità dei valori memorizzabili.

Ricordatevi, comunque, che la memoria a disposizione di AmigaBasic non dipende unicamente dalla disponibilità di memoria Ram, ma anche dallo spazio assegnato al Basic tramite l'istruzione **Clear** (vedi riquadro).

Save

Con questa opzione salvate tutti i dati in memoria. Questi sono registrati in formato ASCII, e quindi facilmente modificabili anche servendovi di un comune Wordprocessor (purchè questo sia in grado di registrare files nel formato Ascii); attenzione, comunque, a digitare i dati in modo corretto.

L'istruzione Clear

L'istruzione Clear corrisponde, in (minima) parte, all'istruzione **Clr** del basic del C/64. Impartendo Clear da programma, o in modo diretto, si cancellano le variabili; con AmigaBasic, invece, Clear può precedere due espressioni...

CLEAR [,areadatibasic] [,areastack]

...come potete controllare sul manuale. La prima espressione **[,areadatibasic]** permette di variare la quantità di memoria a disposizione dell'interprete. Per gestire una matrice tanto grande da richiedere molta memoria Ram, possiamo operare in due modi: in modo diretto (digitando **Clear, 25000: Clear, 300000**) e poi far partire il programma (300000 è lo spazio che riteniamo necessario riservare al basic. Per un piccolo bug è meglio farlo precedere da **Clear, 25000**); oppure scrivendo un programma, da denominare, ad esempio, con **Boot**, contenente le seguenti istruzioni:

CLEAR, 25000: CLEAR ,300000
CHAIN "nomeprogramma"

...in cui **nomeprogramma** è il nome del programma che richiede maggior quantità di memoria a disposizione. Impartendo **Run** al programma **Boot** (precedentemente memorizzato su dischetto) questo assegnerà una porzione maggiore di memoria al Basic e caricherà, subito dopo, il programma *"nomeprogramma"* che verrà mandato in esecuzione.

Se, a questo punto, fermate il programma ed impartite **List**, noterete che in memoria è presente non più il programma Boot ma, appunto, "nomeprogramma". La procedura descritta evita, ogni volta che volete far partire *"nomeprogramma"*, di scrivere in modo diretto i due comandi.

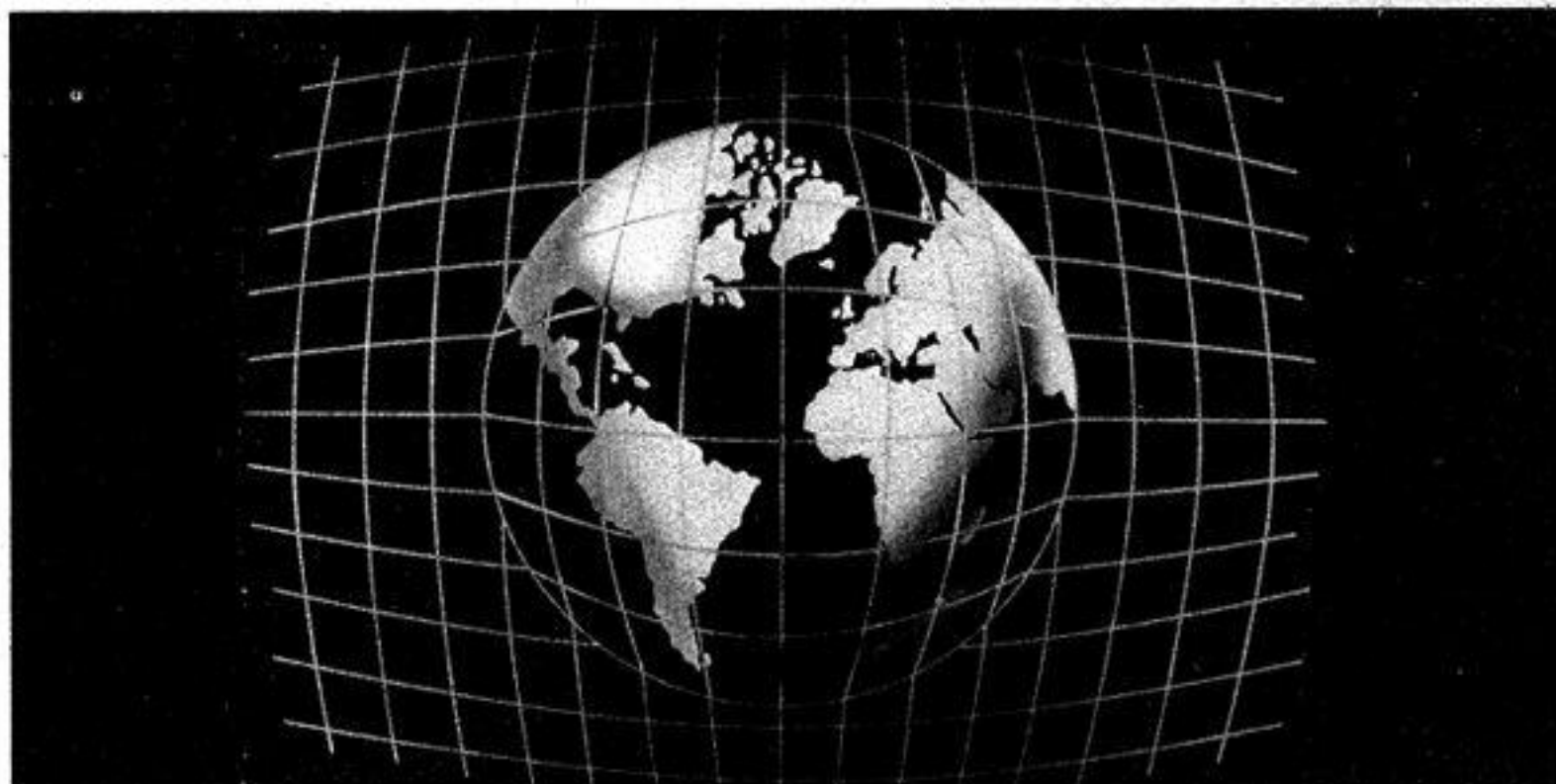
Elenco e Visualizza

Dopo aver inserito dei dati (tramite introduzione / modifica oppure load) è possibile vederne l'elenco o visualizzare una cartina tra quattro modalità diverse: **azimutale polare stereografica; az. pol. equivalente; az. pol. centrografica; cilindrica equatoriale di Mercatore**. Si consiglia di usare l'opzione "elenco" per controllare la correttezza dei dati inseriti; per correggerli, invece, bisogna ricordare il codice del dato errato e modificarlo tramite l'opzione **introduz. / modifica**.

Con l'opzione **visualizza**, dopo aver indicato il tipo di cartina voluto, potete decidere se far apparire sullo schermo, accanto al "segnalino" (micro-rettangolo lampeggiante) il nome della località corrispondente. In ogni caso, in alto sul monitor, compaiono i dati della località. Il segnalino della località visualizzata lampeggerà fino alla pressione del tasto di **spazio**. Premendo, invece, il tasto **ESC** si interromperà la visualizzazione.

Le città italiane

Per "gustare" subito il programma, proponiamo la lista di una quarantina di città che potrete copiare in toto o parzialmente. La scelta delle città è stata fatta in modo tale che, in fase di visualizzazione, sia facilmente riconoscibile la forma della nostra penisola; precisiamo, inoltre, che i dati sono leggermente approssimati. La lista presente nel riquadro non deve, ovviamente, considerarsi limitata: atlante alla mano, potrete aggiungere tutte le località che desiderate.

Località	Long.	Lat.
Aosta	7.3	45.7
Agrigento	13.35	37.18
Ancona	13.31	43.37
Arezzo	11.52	43.27
Ascoli Piceno	13.35	42.51
Bari	16.52	41.07
Bolzano	11.20	46.30
Brindisi	17.56	40.38
Cagliari	9	39.13
Catania	15.05	37.30
Catanzaro	16.35	38.54
Cesena	12.14	44.08
Cosenza	16.15	39.17
Cremona	10	45.08
Cuneo	7.32	44.23
Ferrara	11.36	44.5
Firenze	11.13	43.46
Foggia	15.31	41.27
Genova	8.56	44.25
Grosseto	11.06	42.45
La Spezia	9.46	44.06
Lecce	18.10	40.31
Livorno	10.18	43.33
Merano	11.2	46.7
Messina	15.33	38.1
Milano	9.11	45.29
Napoli	14.16	40.51
Palermo	13.21	38
Pesaro	12.54	43.54
Pescara	14.12	42.27
Potenza	15.48	40.38
Ragusa	14.43	36.55
Roma	12.29	41.53
San Remo	7.46	43.49
Sassari	8.33	40.43
Sondrio	9.8	46.2
Trapani	12.32	38
Trieste	13.46	45.38
Udine	13.14	46.03
Venezia	12.19	45.26

Le principali elaborazioni svolte dal programma sono: apertura di una finestra; richiesta sul tipo di cartina e sulla visualizzazione dei nomi delle località; calcolo degli estremi della cartina per lo sfruttamento ottimale dello schermo; disegno dei contorni, abilitazione della funzione di ricerca da menu e visualizzazione delle località.

Ricerca

Questa opzione è selezionabile solo dopo aver scelto "visualizza cartina"; permette di far lampeggiare la località ricercata e di visualizzarne i dati. E' possibile ricercare una località fornendone il nome completo o parziale. Per esempio, per individuare Milano si può inserire, indifferentemente, **Milano**, milano, **mil**, no, **ILa**. Per ottenere una ricerca così facile sono state utilizzate le funzioni **UCASE$** e **INSTR** (vedi listato, in sezione **ricerca**) che, rispettivamente, permettono di trasformare in maiuscolo una stringa e di cercare una stringa all'interno di un'altra (e non solo agli estremi come con **Left$** e **Right$**).

Le formule matematiche

Riportiamo, nell'apposito riquadro, le relazioni matematiche (relative ai vari tipi di cartine selezionabili) che convertono latitudini e longitudini in coordinate **x**, **y**. Il problema, però, non è la conversione, ma l'uso ottimale delle coordinate x, y allo scopo di evitare il calcolo di valori troppo alti.

Primo gruppo
```
c = 2 * TAN (pi / 4 - lat# / 2) * d
x = - c * SIN (lon#)
y = - c * COS (lon#)
```

Secondo gruppo
```
c = 2 * (1 - SIN (lat#) ) * d
x = - c * SIN (lon#)
y = - c * COS (lon#)
```

Terzo gruppo
```
c = TAN (pi / 2 - lat#) * d
x = - c * SIN (lon#)
y = - c * COS (lon#)
```

Quarto gruppo
```
c = LOG (TAN (pi/4 - lat#/2))
x = - lon# * d / (2 * pi)
y = - c * d / (2 * pi)
```

Le relazioni matematiche

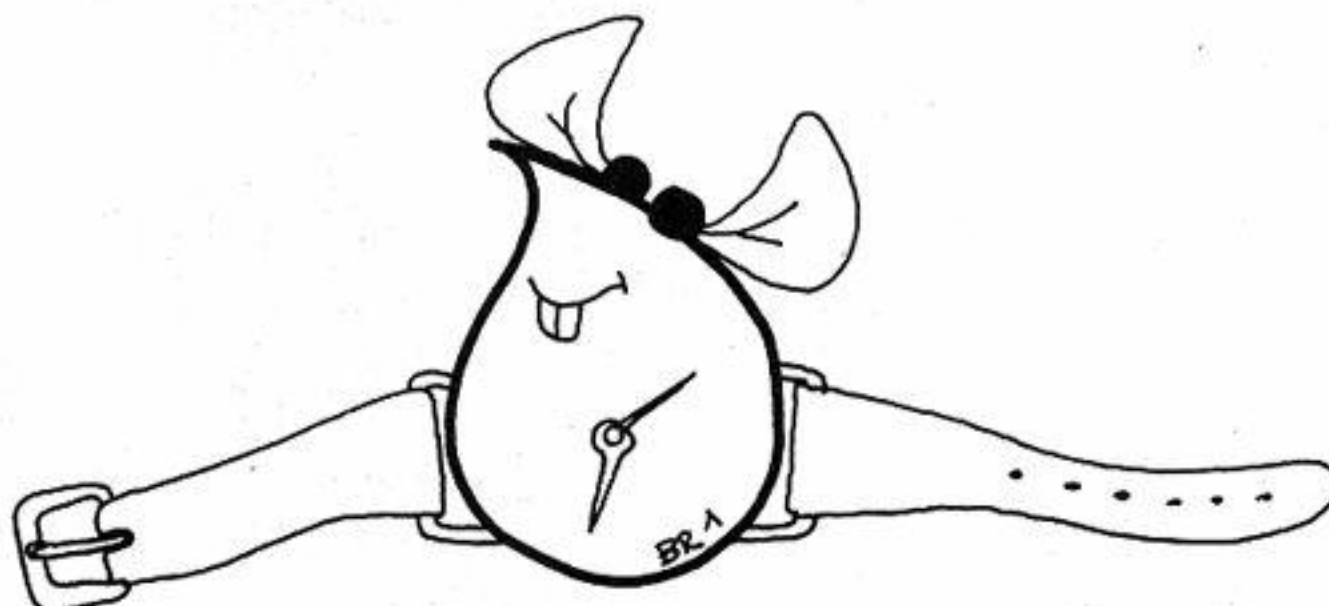
Ecco i quattro gruppi di formule matematiche adoperate nel listato. In queste, **lon#** *e* **lat#** *indicano longitudine e latitudine;* **pi** *= 3.14;* **d** *= raggio / scala;* **raggio** *è quello terrestre nell'esempio considerato,* **scala** *quella desiderata. Il programma proposto calcola automaticamente la scala migliore.*

```
' rem Cartina geografica terrestre, lunare o stellare
' rem per Amiga (Preferences = 80 colonne) by Spataro Valentino
max=500:DIM posto$(max),lon#(max),lat#(max)
CLS:pi=3.14159266#:conv=pi/180:pianeta$="Terra":r=6371000&
minlon#=360:maxlon#=0:minlat#=360:maxlat#=-360
DATA 1,0,1,"project",1,1,1,"new",1,2,1,"load",1,3,0,"save"
DATA 1,4,1,"directory",2,0,1,"cartina",2,1,1,"introduzione-modifica"
DATA 2,2,0,"elenco",2,3,1,"default",2,4,0,"visualizza cartina"
DATA 2,5,0,"ricerca",3,0,0,"",4,0,0,"",1,5,1,"quit",0,0,0,""
1 READ a1,a2,a3,a$:IF a1<>0 THEN MENU a1,a2,a3,a$:GOTO 1
ON MENU GOSUB scelta:MENU ON:REM *** fine inizializzazioni ***
main:LOCATE 1,1:PRINT SPACE$(78)
m1:LOCATE 1,1:PRINT "Scegli da menu'.";w%"dati in memoria":GOTO m1
scelta:ON MENU(0) GOTO progetto,gestione
progetto:ON MENU(1) GOTO start,loaddati,savedati,dir,fine
gestione:ON MENU(1) GOTO inputdati,elenco,default,cartina,ricerca
start:a$="confermi ?":conferma a$:IF a$="S" THEN RUN ELSE RETURN
fine:a$="confermi ?":conferma a$:IF a$="N" THEN RETURN ELSE MENU RESET:END
inputdati:WINDOW 2,"introduzione/modifica":PRINT  w%"dati in memoria"
  PRINT  "codice da modificare ("w%+1"=aggiunge)":INPUT cod
  IF cod=0 THEN WINDOW CLOSE 2:RETURN
  IF cod>max THEN PRINT "fine memoria":GOSUB aspetta:WINDOW CLOSE 2:RETURN
  INPUT "denominazione localita' ";a1$:IF a1$="" THEN a1$=posto$(cod)
  INPUT "longitudine ";a2#:a2#=a2#*conv:IF a2#=0 AND a4<>0 THEN a2#=lon#(a4)
  INPUT "latitudine ";a3#:a3#=a3#*conv:IF a3#=0 AND a4<>0 THEN a3#=lat#(a4)
  a$="confermi ?":conferma a$:IF a$="N" GOTO inputdati
  IF minlon#>a2# THEN minlon#=a2#
  IF maxlon#<a2# THEN maxlon#=a2#
  IF minlat#>a3# THEN minlat#=a3#
  IF maxlat#<a3# THEN maxlat#=a3#
  IF cod=w%+1 THEN w%=w%+1
  posto$(cod)=a1$:lon#(cod)=a2#:lat#(cod)=a3#
  MENU 1,2,1:MENU 2,2,1:MENU 1,3,1:MENU 2,4,1:GOTO inputdati
elenco:WINDOW 2,"elenco dati":' visualizza tutti i dati in memoria
  CLS:PRINT "premi barra spazio per visualizzare,ESC per uscire"
  FOR a=1 TO w%:a$=""
    WHILE a$<>" " AND a$<>CHR$(27):a$=INKEY$:WEND
    IF a$=CHR$(27) THEN a=w% :ELSE GOSUB datilocalita
  NEXT:aspetta:WINDOW CLOSE 2:RETURN
'***** REM PARTE RELATIVA ALLA GESTIONE DEL DRIVE *****
dir:WINDOW 2,"dir":INPUT "path ";a$:FILES a$:aspetta:WINDOW CLOSE 2:RETURN
loaddati:WINDOW 2,"load dati":f=0:'************
 INPUT " nome file";f1$:IF f1$="" THEN WINDOW CLOSE 2:RETURN
  ON ERROR GOTO 1300:'rem attiva la rivelazione di errori
  a$="Tieni i dati che hai in memoria ?":conferma a$:IF a$="N" THEN w%=0
 OPEN f1$+".geo" FOR INPUT AS 1:IF f=1 THEN f=0:GOTO loadfine
  INPUT#1,pianeta$:INPUT#1,r
  IF a$="N" THEN minlon#=180:maxlon#=-180:minlat#=90:maxlat#=-90
 WHILE NOT EOF(1):w%=w%+1:'per NOT EOF(1) vedi articolo
  IF w%>max THEN PRINT "MEMORY FULL. ":GOTO loadfine
  INPUT#1,posto$(w%),a1#,a2#
  a1#=a1#*conv:a2#=a2#*conv:lon#(w%)=a1#:lat#(w%)=a2#
  IF minlon#>a1# THEN minlon#=a1#:' durante il caricamento calcola le
  IF maxlon#<a1# THEN maxlon#=a1#:' minime e massime latitud. e longitud.
  IF minlat#>a2# THEN minlat#=a2#:' utili per calcolare poi nella
  IF maxlat#<a2# THEN maxlat#=a2#:' visualizzazione la scala migliore.
```

```
 WEND:' loop di caricamento dati. se f=1 allora si e' verificato un errore
  IF f<>1 THEN MENU 1,2,1:MENU 2,2,1:MENU 1,3,1:MENU 2,4,1
loadfine:CLOSE 1:ON ERROR GOTO 0:WINDOW CLOSE 2:RETURN
savedati:f=0:WINDOW 2,"save dati":'***********
  INPUT " nome file";fl$:IF fl$="" THEN WINDOW CLOSE 2:RETURN
  ON ERROR GOTO 1300
  OPEN fl$+".geo" FOR OUTPUT AS 1:IF f=1 THEN f=0:GOTO savefine
  PRINT#1,pianeta$:PRINT#1,r
  FOR a=1 TO w%
   PRINT#1,posto$(a):PRINT#1,lon#(a)/conv:PRINT#1,lat#(a)/conv
  NEXT
savefine:CLOSE 1: ON ERROR GOTO 0:WINDOW CLOSE 2:RETURN
cartina:WINDOW 2,"cartina":CLS:'**************
  PRINT "quale cartina:":PRINT "1 azimutale polare stereografica"
  PRINT "2 azimut. pol. equivalente":PRINT "3 azimut. pol. centrografica"
  PRINT "4 cilindrica equatoriale di Mercatore":PRINT "5 menu'"
  cart=0:WHILE (cart=0 OR cart>5):cart=VAL(INKEY$+CHR$(0)):WEND
  IF cart=5 THEN WINDOW CLOSE 2:RETURN:ELSE PRINT ">"cart
  a$="Vuoi stampati i nomi delle citta' ?":conferma a$:l$=a$:WINDOW CLOSE 2
 REM CALCOLA SCALA MIGLIORE E CENTRO (X1,Y1)
   x1=0:y1=0:scala=1:d=r/scala
   lat#=minlat#:lon#=minlon#:GOSUB 1200:miny=y:minx=x:' calcolo estreme
   lat#=maxlat#:lon#=maxlon#:GOSUB 1200:maxy=y:maxx=x:' coordinate
   sx=(maxx-minx)/600:sy=(maxy-miny)/320:' calcolo scale ottimali per assi
   x1=-((maxx+minx)/2):y1=-((maxy+miny)/2):'calcolo coord. del centro
   IF sx>sy THEN scala=sx:ELSE scala=sy:'tra le due scale prende la maggiore
   CLS:d=r/scala:x1=x1/scala+320:y1=y1/scala+100
 REM DISEGNA I CONTORNI DELLA ZONA
  LOCATE 1,36:PRINT "Scala= 1:"scala
  FOR lon#=minlon# TO maxlon# STEP pi/360
   lat#=minlat#:GOSUB 1200:lat#=maxlat#:GOSUB 1200
  NEXT:FOR lat#=minlat# TO maxlat# STEP pi/360
   lon#=minlon#:GOSUB 1200:lon#=maxlon#:GOSUB 1200
  NEXT
 REM VISUALIZZA LE LOCALITA'. SE L$="S" STAMPA ACCANTO LE DENOMINAZIONI
  LOCATE 1,1:a$="Dati esatti ?":conferma a$:IF a$="N" GOTO cartina
  MENU 2,5,1:REM attiva la ricerca, che puo' essere fatta solo dopo cartina
  LOCATE 23,1:PRINT "premi spazio per proseguire, ESC per finire";
  FOR a=1 TO w%:
   lat#=lat#(a):lon#=lon#(a):GOSUB 1200:REM conversioni lat/lon in x,y
   LOCATE 1,1:PRINT SPACE$(79):luogo nx,ny,1!:LOCATE 1,1:GOSUB datilocalita
   IF l$="S" AND ny/8+1<24 AND nx/8+1<79 THEN
     LOCATE ny/8+1,nx/8+1:PRINT posto$(a);:a$=""
   ELSE
     GOSUB lampeggia
   END IF:WHILE a$<>" " AND a$<>CHR$(27):a$=INKEY$:WEND
   IF a$=CHR$(27) THEN a=w%
  NEXT:COLOR 3:aspetta:COLOR 1:RETURN
default:WINDOW 2,"default":'**************
  PRINT pianeta$;:INPUT "pianeta ";a$:IF a$="" THEN a$=pianeta$
  PRINT r;:INPUT "raggio ";a1:IF a1=0 THEN a1=r
 a$=pianeta$:r=a1:WINDOW CLOSE 2:RETURN
ricerca:LOCATE 1,1:s$=SPACE$(79):PRINT s$:LOCATE 1,1:h=0
  INPUT "nome citta' da cercare (anche parziale) ";r$:IF r$="" THEN RETURN
  FOR a=1 TO w%
    IF INSTR(UCASE$(posto$(a)),UCASE$(r$))<>0 THEN h=a:a=w%
```

```
  NEXT:IF h=0 THEN PRINT "non trovato":aspetta:GOTO ricerca
  c=0:a=h:LOCATE 23,1:PRINT "premi un tasto ";
  lat#=lat#(a):lon#=lon#(a):GOSUB 1200:LOCATE 1,1:PRINT s$
  LOCATE 1,1:GOSUB datilocalita:a$="":GOSUB lampeggia:LOCATE 1,1:PRINT s$
RETURN

REM --------FINE PROGRAMMA. INIZIO SEZIONE NUOVI COMANDI -----------------
' VISUALIZZA UN QUADRATO DI COORD. NX,NY. COL E' IL COLORE
SUB luogo (nx,ny,col) STATIC
  LINE (nx-1,ny-1)-(nx+1,ny+1),col,bf
END SUB
' CONFERMA PERMETTE DI RISPONDERE S/N A UN MESSAGGIO VARIABILE
' A$ ALL'INIZIO CONTIENE IL MESSAGGIO, IN USCITA LA RISPOSTA IN MAIUSCOLO
SUB conferma (a$) STATIC
  PRINT a$" ";:a$=""
  WHILE (a$<>"S" AND a$<>"N"):a$=UCASE$(INKEY$):WEND:PRINT a$
END SUB
SUB aspetta STATIC
  LOCATE 23,1:PRINT "premi tasto sinistro del mouse per proseguire";
  a=MOUSE(0):WHILE MOUSE(0)>=0 :WEND:LOCATE 23,1:PRINT SPACE$(60);
END SUB
REM ---------FINE SEZIONE NUOVI PROGRAMMI. INIZIO SUBROUTINES ------------
' VISUALIZZA I DATI RELATIVI AD UNA DATA LOCALITA'
datilocalita:
  PRINT pianeta$"#"a"/"w%;posto$(a)TAB(40)" lon";
  PRINT USING " ###.### ";lon#(a)/conv;
  PRINT "lat";:PRINT USING " ###.###";lat#(a)/conv:RETURN
lampeggia:'la subroutine lampeggia fa lampeggiare il luogo desiderato
  a$="":c=0:
  WHILE a$="":FOR c=1 TO 100:NEXT:IF col=1 THEN col=0 ELSE col=1
    luogo nx,ny,col:a$=INKEY$
  WEND:luogo nx,ny,1!:RETURN
' QUI AVVIENE LA CONVERSIONE LAT/LON IN X/Y A SECONDA DELLA CARTINA SCELTA
' PRIMA DEL GOSUB IMPOSTARE LAT# E LON#; VERRANNO RITORNATE NX,NY PER PSET
1200 REM FORMULE MATEMATICHE DI CONVERSIONE
     ON cart GOSUB 1210,1220,1230,1240
     x=x1+x:y=y1+y:IF ABS(x)>32765 OR ABS(y)>32765 THEN RETURN
     nx=600-x:ny=(288-y)*.44:PSET (nx,ny):RETURN
1210 c=2*TAN(pi/4-lat#/2)*d:x=-c*SIN(lon#):y=-c*COS(lon#):RETURN
1220 c=2*(1-SIN(lat#))*d:x=-c*SIN(lon#):y=-c*COS(lon#):RETURN
1230 c=TAN(pi/2-lat#)*d:x=-c*SIN(lon#):y=-c*COS(lon#):RETURN
1240 c=LOG(TAN(pi/4-lat#/2)):x=-lon#*d/(2*pi):y=-c*d/(2*pi):RETURN
' SUBROUTINE PER LA RILEVAZIONE DEGLI ERRORI
1300 f=1:CLS:PRINT "errore numero:"ERR:aspetta:RESUME NEXT
```

POSTAMIGA

(a cura di Domenico Pavone)

Attenzione, pericolo!
Mi si è rotto il mouse, e vorrei sapere se posso adoperare, per il mio Amiga 2000, quello che mio padre usa sul suo IBM.
(Marco Varzini - Venezia)

Assolutamente no! I contatti al connettore nei due tipi di mouse sono differenti, e un loro uso indiscriminato può provocare danni anche molto gravi al computer, a seconda della versione di 2000 posseduta.

E per molto gravi si intendono proprio effetti come un angoscioso odore di bruciato, cui farà inevitabilmente seguito un non indifferente esborso pecuniario a vantaggio del più vicino centro di assistenza...

Workbench disordinato
Ho da poco un Amiga 500, e non riesco a rimettere in ordine le icone contenute all'interno di una directory. O meglio: adoperando Snapshot dal Workbench, dopo avere clickato tutte le icone con lo shift premuto (come suggerito dal manuale) le icone poi restano in ordine, ma la finestra che le contiene risulta troppo piccola o troppo grande, mai come l'avevo sistemata io prima dello Snapshot.
(Vincenzo Petra - Napoli)

Evidentemente la (giustificata) foga di "provare", ha impedito una più approfondita lettura del manuale. L'operazione di **Snapshot**, così come descritta, è grosso modo corretta, tranne per un dettaglio: se si intende riordinare in modo **permanente** il contenuto di una directory (lo stesso vale per una icona-disco), va sì dimensionata la finestra, vanno clickati con lo shift abbassato le icone presenti al suo interno, ma va **anche** clickata (sempre con lo shift premuto) l'icona della directory!

Anche indipendentemente dalle icone, le dimensioni della finestra vanno collegate alla sua directory clickando prima sull'icona-cassetto (se non ridisegnata, s'intende), poi nella finestra già dimensionata.

Quest'ultima operazione "attiva" la finestra, cosa che avviene automaticamente se si agisce sulle icone in essa contenute.

E' buona prassi, in definitiva, clickare **PRIMA** nell'icona della directory; poi, con lo shift abbassato, nella finestra adattata alle dimensioni che si desiderano e sulle sue icone, infine selezionare Snapshot dal menu Special di Workbench.

File bloccato?
Programmando in assembler, mi è capitato un problema che non riesco a risolvere. Ho utilizzato le routine del FIB (Lock, Examine, Exnext) per esaminare un file, ma dopo non riesco più a cancellarlo, modificarlo o altro. Come posso evitare questo inconveniente?
(Gabriele Motteran - Bussolengo)

Sulla base della sola lettera, che non contiene alcun disassemblato di riferimento, non è facile dare una risposta esauriente, in quanto potrebbe anche trattarsi di un banale errore di programmazione. Il motivo più probabile di tale comportamento, co-

munque, potrebbe essere imputato ad un mancato "sbloccaggio" del file in esame. Le routine che coinvolgono il **FIB** (**F**ile **I**nfo **B**lock, trattato su Postamiga del **n. 77**), comprendono tutte la richiesta di un **Lock** nei confronti del file o directory da esaminare. Per chi non fosse molto esperto nella programmazione Assembly, limitiamoci in questa sede a ribadire che la sintassi della funzione Lock (della **Dos.library**) prevede che gli si inviino, come parametri, l'indirizzo ove è memorizzata la stringa specificante il nome del file (nel registro **D1**) e il modo di accesso (in **D2**). Di ritorno, si avrà in **D0** quello che, appunto, definiamo Lock. Il secondo parametro, in particolare, può essere impostato come Accesso in Lettura o Accesso in Scrittura (quest'ultimo, tra l'altro, ne rende esclusivo l'accesso).

Rimandando al n. 77 per un esempio pratico di applicazione, ricordiamo al nostro lettore che, una volta esaminati i dati inviati alla struttura FIB dalle funzioni **Examine** oppure **ExNext**, è indispensabile richiamare la funzione **Unlock**, per rimuovere il Lock prima creato.

Niente di più facile che il problema stia proprio qui: un mancato sbloccaggio del file, vuoi per non aver richiamato Unlock, vuoi per un qualche errore che non fa giungere il flusso del programma alla eventuale chiamata di questa funzione, potrebbe essere all'origine dell'inconveniente.

Driver, eterno problema

Posseggo una stampante Star acquistata (su consiglio del rivenditore) assieme al mio Amiga 500, che utilizzo tramite il drive Epson JX-80. Il tutto funziona egregiamente per i testi, ma non altrettanto con la stampa grafica. Esiste un driver specifico per questa stampante?
(Paolo Fiore - Benevento)

Purtroppo non abbiamo sufficiente confidenza con quella marca di stampante per una risposta più esauriente. Possiamo solo consigliare di provare anche il driver **EpsonXOld**, sempre che l'emulazione Epson sia totalmente implementata... e che sia possibile la stampa grafica.

Maggiori informazioni, in tal senso, dovrebbero essere presenti sul manuale della stampante. Un consiglio per l'acquisto: se non si ha già una certa esperienza e ci si rivolge alla fascia media delle *9 aghi*, perché non evitare ogni problema affidandosi a stampanti che portano lo stesso marchio di fabbrica di Amiga (Commodore, insomma)?

Amiga o Pc?

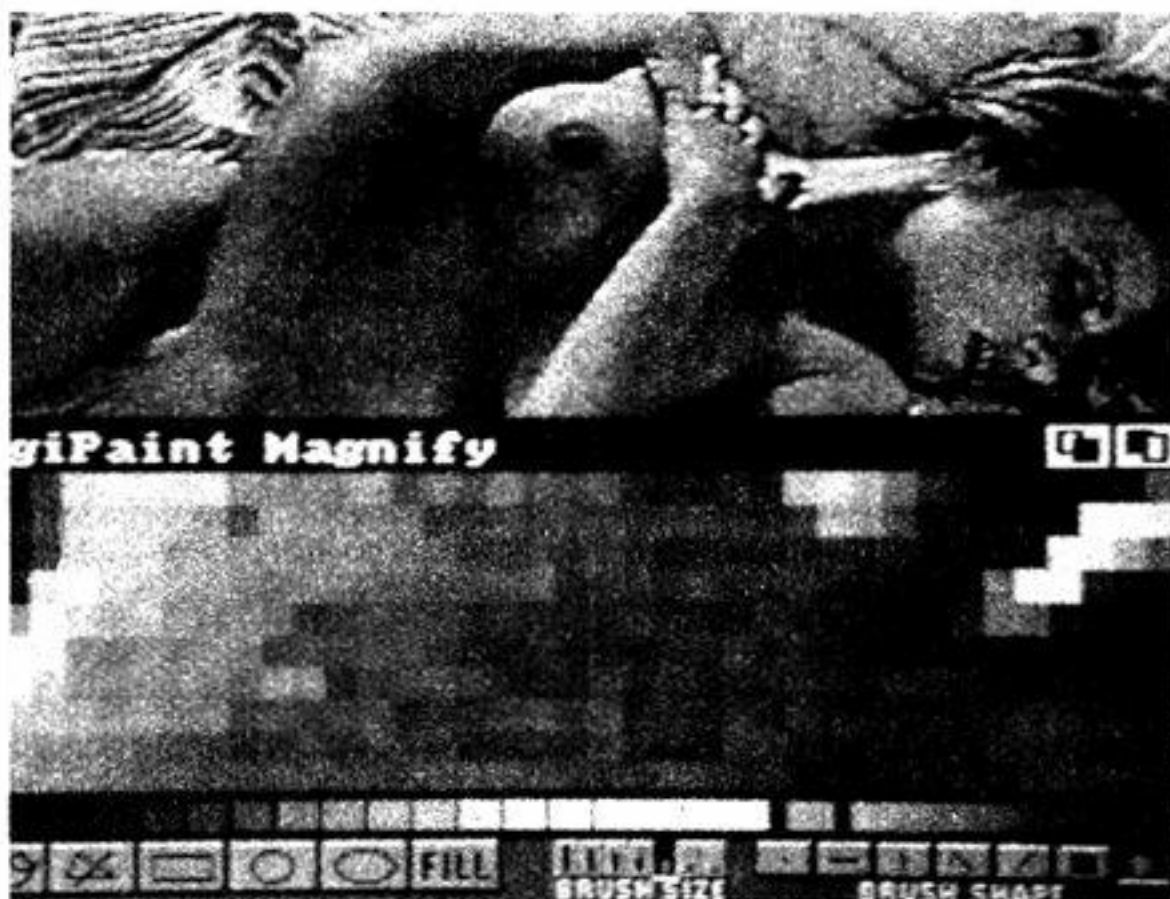
Vorrei passare ad un sistema più potente del mio attuale (C/64), ma sono indeciso tra un Amiga o un Ibm compatibile. I miei genitori possiedono una piccola azienda e mi piacerebbe prendere qualcosa che possa servire anche per essa, ma non vorrei rinunciare a qualche partita con i videogiochi, e so che l'Amiga è molto forte in questo campo. Se prendo Amiga, ci sono poi programmi per utilizzarlo anche in azienda? Un'altra domanda: è vero che non viene più prodotto l'Amiga 2000?
(Silvano Fabris - ??)

Arduo dilemma... e ardua risposta. Nel senso che il consiglio più ovvio sarebbe quello di acquistare entrambi i tipi di computer, e non rinunciare a niente. Ma questo tipo di soluzione è difficilmente proponibile, a meno che non si abbia una disponibilità economica che lo consenta. In teoria, con Amiga ci si può fare quello che si vuole: non mancano certo programmi di archiviazione o fogli elettronici; ma, se si parla di azienda... il discorso si fa scabroso.

In tutta sincerità, ed a titolo puramente personale (nonché rischiando un linciaggio a furor di popolo amigo), la scelta più ovvia non può che essere un Pc, anche se a discapito di qualche *game* di pregiata fattura che **solo** Amiga riesce a far godere. Comunque, non è che poi i videogiochi siano totalmente assenti in ambiente Ms - Dos, solo bisogna accontentarsi...

Il motivo del consiglio è presto detto: con Amiga sarebbe necessario (fino ad ora, almeno) molto "far da sé", nel senso di adattare i programmi esistenti (da non disprezzare, si badi bene) alle esigenze di una ditta. Se si è in grado di farlo, allora potrebbe anche andar bene.

Di contro, per i PC compatibili esiste una marea di software dedicato espressamente a fini gestionali, non c'è che l'imbarazzo della scelta, o farsi consigliare dall'altrettanto elevato numero di consulenti, programmatori, eccetera.

Un'altra considerazione riguarda, poi, l'hardware ed i relativi costi. Per applicazioni professionali (anche escludendo il settore grafico-musicale, ove Amiga può risultare invece una scelta ottimale), sarebbe in ogni caso necessario rivolgersi a configurazioni di un certo impegno (hard

disk, memoria, modelli superiori al 500, eccetera), che in definitiva costano parecchio più che le corrispondenti in ambiente Pc.

La conclusione, insomma, è più che chiara...

Quanto agli Amiga 2000, la diceria è decisamente infondata: godono di ottima salute, e non si prevede al momento una loro scomparsa dal mercato.

Con un po' di P.D.

Quando avevo un C/128 adoperavo un sistema di protezione dei programmi su disco basato su una parola d'ordine. Vorrei qualcosa di simile per Amiga, ma sembra non esista...
(D. Militello - Villabate)
Dove è possibile trovare i nomi e i parametri di chiamata delle funzioni della Arp Library?
(Giacomo Zari)

Pur se di diverso tenore, le due risposte possono essere ricondotte ad un'unica, straripante fonte: il materiale di **Pubblico Dominio**, la cui reperibilità non dovrebbe porre alcun problema, grazie anche ad una ormai ampia diffusione del mezzo telematico (le varie BBS p. es. della rete Fido, per intenderci). Se anche non si possedesse un modem, si può sempre attingere alle varie raccolte rintracciabili nei negozi, o ancora cercare qualche scambio spulciando gli annunci delle varie riviste.

In particolare, di programmi che limitano l'accesso a files o dischi a sole persone autorizzate ne esistono parecchi, tutti con nomi piuttosto simili. Citiamo ad esempio **Code**, la cui sintassi (da **Shell** oppure **Cli**) è autoesplicativa:

Code <inputfile> <password> <outputfile> [decode]

Chi non amasse le finestre di Shell, può invece rivolgersi a **File Coder**, di cui circolano più versioni, alcune delle quali dotate di finestre-menu per facilitare le operazioni. Per qualcosa di più globale, che coinvolge l'intero dischetto, si può sfruttare **Disk Coder**, sempre circolante in ambienti più o meno PD. Tra non molto, comunque, qualcosa di simile verrà pubblicato anche nelle pagine della nostra rivista.

Quanto alla **Arp Library**, questa di solito viene distribuita in "archivi" compattati comprendenti files di testo-documentazione. Chi, come il nostro lettore, è interessato alla programmazione, può trovare quanto cerca in files (compattati) di nome **Arppro.xxx** o similari, con **xxx** che può corrispondere a **zoo**, **arc**, **lzh** oppure **zip**, a seconda del compattatore adoperato (l'originale è in formato zoo). Se proprio non si riuscisse a trovarlo, si può tentare di contattare per posta la Microsmiths Inc. all'indirizzo P.O. Box 561 - Cambridge, MA 02140.

Ma che memoria è?

Ho notato uno strano comportamento del mio Amiga 2000 con alcuni programmi. In particolare, non riesco a far funzionare le animazioni con Deluxe Paint III. Ho pensato che fosse dovuto al programma non originale, ma lo stesso poi funziona nell'Amiga 500 espanso ad 1 Megabyte di un mio amico. Com'è possibile? Forse il mio computer non si "accorge" di avere i 512 Kram in più, come il 500 espanso del mio amico? Devo utilizzare il 2000 per un falò in giardino?
(Marco Frazzoli - Montecatini)

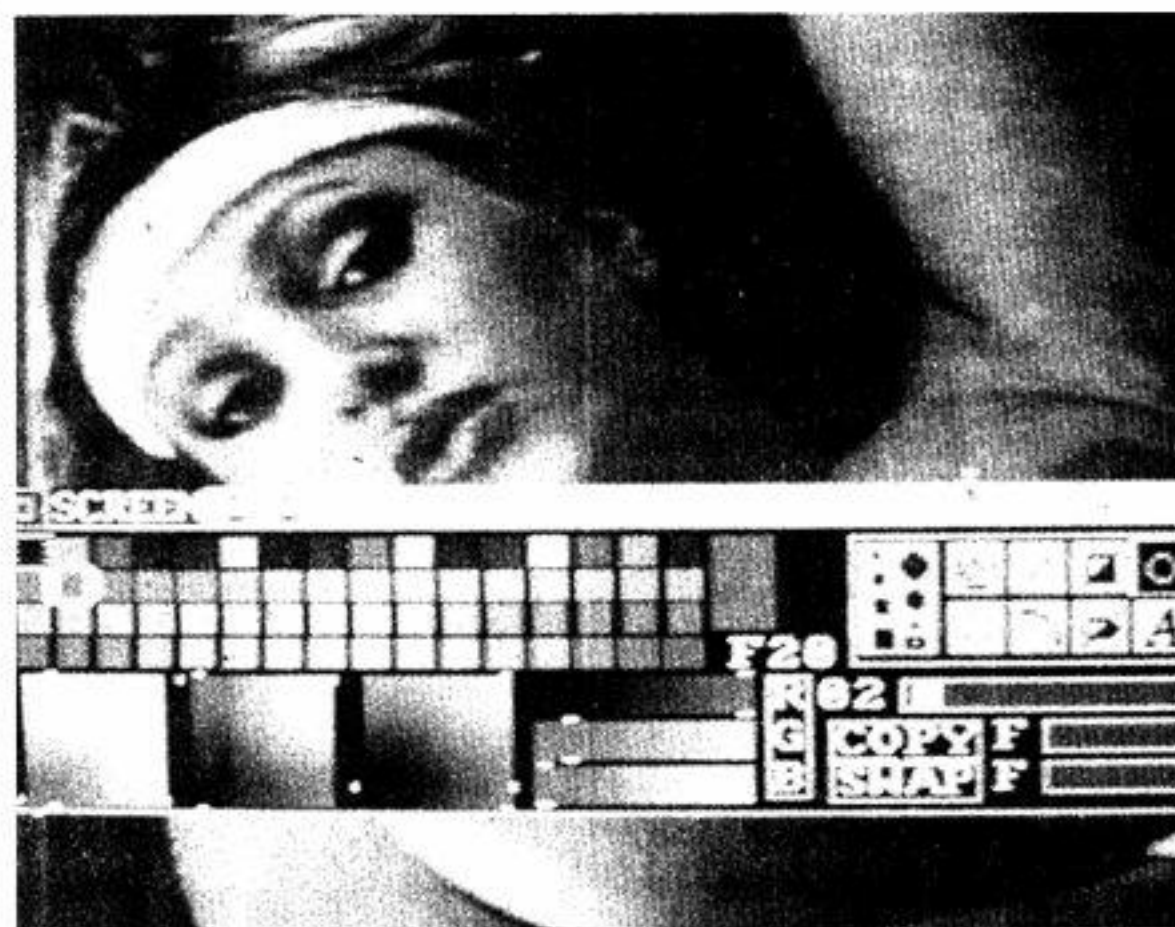
Il motivo della disparità di comportamento potrebbe anche essere dovuto a qualche malfunzionamento hardware, come sospetta il nostro lettore nella sua accorata (e solo parzialmente riportata) lettera. Ma è anche possibile che la causa sia più banalmente dovuta ad una differente configurazione della memoria. Anche se le due "macchine" adoperate per far girare lo stesso programma hanno 1 megabyte di memoria, non è detto che siano identiche.

Prima di abbandonarsi alla disperazione, è forse il caso di controllare se il 500 in questione non sia un modello **più recente** del 2000, soprattutto in rapporto alla eventuale presenza del **nuovo Agnus**, in grado di vedere l'intero megabyte come memoria **Chip**. In questo caso, potrebbe benissimo darsi il caso che il 2000, dotato di soli 512 K di Chip Ram (massicciamente utilizzata nelle applicazioni grafiche), non sia in grado di attivare qualche animazione particolarmente "robusta", ovvero dotata di parecchi frame. Al contrario sul 500, grazie al Mega, tutto di chip Ram, la stessa animazione potrebbe girare correttamente.

Naturalmente è un'ipotesi, eventualmente da comprovare ricorrendo al comando **Avail** da ambiente **Shell** per verificare se la Ram disponibile sui due computer è effettivamente diversa.

Prima di passare ai falò.

L'ebbrezza del 2.0...

Mi sono "procurato" la versione 2.0 del Workbench, per provarla sul mio Amiga

500. Però il computer si rifiuta di farmelo vedere, segnalandomi che la versione del kickstart non è quella giusta. Vuol dire che il 2.0 sarà riservato solo a chi acquista un Amiga 3000?
(Carlo Bonardi - Firenze)

Fino ad ora, da un punto di vista strettamente hardware, sì.

Nel senso che il **Workbench 2.0** richiede sia presente il nuovo **Kickstart 2.0**, di dimensioni **raddoppiate** rispetto ai precedenti (512 KB), e che fino al momento in cui si scrive non è "upgradabile" (brutto neologismo, vero?) sui modelli 500 e 2000. Si parla di future implementazioni che richiederanno, come ovvio, un adeguamento hardware delle Rom, limitatamente al Kickstart, ma è tutto ancora di là da venire.

Comunque, giusto per dare un'occhiata e provare l'ebbrezza di aggirarsi tra le icone del nuovo Workbench, è possibile attivarlo anche sugli Amiga 500 e 2000, a patto che disponga di Ram sufficiente, vale a dire al di sopra di 1 Megabyte. Naturalmente è poi necessario un artifizio software, ma niente di particolarmente complesso. Esistono, infatti, nell'ambito del Pubblico Dominio ben due programmi (di nome **KickIt** e **ZKick**) in grado di caricare in Ram il Kickstart 2.0, ed operare i necessari cambiamenti per far credere al sistema che sia questo il suo Kickstart attivo.

Oltre ai programmi, però, è anche necessario disporre di un file contenente il Kickstart vero e proprio, o meglio una delle sue versioni caricabili anche in ambiente diverso dal 3000. Questo non può logicamente essere di pubblico dominio, ma è molto facile ritracciarlo, ne circolano a bizzeffe (del resto è in fondo innocuo provare a far andare il 2.0 su un 500/2000).

Dopo l'installazione software, si ha di norma un reset del computer, con la comparsa del nuovo logo iniziale (animato) in sostituzione dell'orribile vecchia mano-con-disco, e può essere lanciato qualunque programma accettato dal sistema operativo 2.0, nuovo Workbench compreso.

A seconda del programma "caricatore" adoperato, dopo il reset si resterà in ambiente 2.0 (con Zkick), oppure si tornerà al "vecchio" 1.3.

Standard di comunicazione piu' diffusi

V21
Trasmissione dati con velocità di **300** baud. In USA corrisponde al **Bell 103**, peraltro **non** compatibile con il V21.

V22
Trasmissione dati a **1200** baud. Anche in questo caso esiste uno standard solo americano, il **Bell 212A**, non compatibile con il nostro V22.

V22bis
Trasmissione dati a **2400** baud, universalmente accettato.

V23
Standard particolare che trasmette a **1200** baud in una direzione e **75** nell'altra. E' adottato da **VideoTel**, e serve in pratica **solo se** si intende usufruire di quel servizio. L'eventuale implementazione del V23 fa lievitare il costo del modem, ed è in ogni caso poi necessario un software di comunicazione particolare per utilizzare Videotel.

V29
Lo citiamo ad abundantiam, in quanto corrisponde alla trasmissione a **9600** baud in half duplex adoperata dai **fax**, non utile per i normali modem.

V32
Degli standard ufficiali, è al momento quello che definisce la maggiore velocità di trasmissione, **9600** baud, anche se è in fase di definizione un **V32bis** che dovrebbe "regolamentare" il traffico a **14400** baud. Oltre alla velocità di trasmissione, il V32 definisce anche altre modalità ad esso legate (p.es. il Full Duplex), che in questa sede non possono importare molto.

V42 e V42bis
A differenza degli standard finora visti, queste sigle non specificano una velocità di trasmissione, ma definiscono degli standard per la cosiddetta correzione d'errore (**V42**) e correzione d'errore con compressione dei dati (**V42bis**). Al momento attuale questo tipo di implementazioni sono di fatto diffuse basate su un diverso standard diviso in varie classi, l'**MNP**, non compatibile con il V42 o V42 bis, che corrispondono all'MNP 4 ed MNP 5 rispettivamente.

Standard... non standard
Nonostande la codifica ufficiale, si è detto che esistono di fatto altri standard, tra l'altro molto diffusi, soprattutto nel settore delle alte velocità. Un esempio tipico è l'**HST** (High Speed Technology), legato ai modem Courier della U.S. Robotics. Questi sono in grado di raggiungere i 14400 baud, a patto però che all'altro capo della linea sia presente un altro HST. In caso contrario, il collegamento è possibile solo in V22bis (2400 baud) per i modelli normali, oppure in V32 (9600 baud) per i modelli Dual Standard.

Stesso discorso per i supermodem della Telebit, i Trailblazer, che adottano uno standard chiamato **PEP** (Packetized Ensemble Protocol). In entrambi i casi, ci si trova comunque ad un livello decisamente alto, anche da un punto di vista... pecuniario.

Modem: Quale standard?

Vorrei acquistare un modem, ma non so bene quale scegliere.
Girando per negozi mi hanno consigliato quasi tutti di acquistarne uno a velocità di 2400 baud, ma poi mi hanno confuso le idee con un sacco di sigle come V22, V22bis, eccetera. Potreste dirmi cosa significano, e se devo tenerne conto per scegliere quale modem comprare?

(Antonio Lafauci - Palermo)

L'elemento primo sul quale basare la scelta, non può che essere il solito: la disponibilità economica. Considerato, però, il notevole abbassamento dei prezzi in questa fascia, in effetti il consiglio di iniziare con un modem a 2400 baud è più che corretto. Velocità inferiori sono ormai sconsigliabili, mentre è pressoché tassativo **non** prendere in considerazione i **300** baud.

Importante che il modem sia **Hayes compatibile**, anche se ormai è una raccomandazione quasi superflua, visto

che praticamente tutti i modem in circolazione lo sono.

Quanto alle sigle, queste identificano alcuni standard per così dire "ufficiali", ovvero omologati da un istituto internazionale che si occupa appunto di questo. L'istituto è il **CCITT**, acronimo di **C**onsultative **C**ommittee on **I**nternational **T**elephone and **T**elegraph. In effetti, oltre a quelli della CCITT, esistono anche altri standard divenuti tali di fatto, soprattutto nel settore delle comunicazioni ad alta velocità e con correzione di errore e/o compressione dei dati, ma non è troppo difficile districarsi tra le sigle.

Per una loro rapida comprensione, si consulti il riquadro pubblicato in queste pagine, che schematizza quanto più possibile l'argomento fornendo brevi note esplicative per gli standard più diffusi, omologati o "di fatto" che siano. Per maggiori dettagli, si rimanda alla lettura delle pagine della rivista espressamente dedicate alla telematica.

Variabili a zonzo

In un mio programma Basic che deve richiamare alcune librerie di sistema, ho adoperato una tecnica prelevata da un vostro listato per ottenere spazio in memoria ove memorizzare una struttura di dati. Ho cioè creato una variabile con indice, ne ho ricavato l'indirizzo con Varptr, e vi ho pokato i valori che mi interessavano. Poi nel programma mi limitavo a "passare" l'indirizzo alla routine di libreria che voleva la struttura, e tutto sembrava funzionare. Ma mi sono accorto che, più avanti nel programma, non venivano più letti gli stessi valori, e addirittura il computer mi andava in guru...

(Francesco Savio - Savona)

```
' DISEGNA UN POLIGONO
' ADOPERANDO LA GRAPHIC
' LIBRARY DI SISTEMA
LIBRARY "graphics.library"
SCREEN 1,320,250,2,1
WINDOW 2,,,0,1
Rast&=WINDOW(8)
DIM a%(16)
FOR x=0 TO 15
 READ y
 a%(x)=y
NEXT
Move& Rast&,100,50
Polydraw& Rast&,8,VARPTR(a%(0))
Move& Rast&,75,95
PRINT "clicka col mouse"
Move& Rast&,104,110
PRINT "per uscire"
loop:
IF MOUSE (0)=0 THEN loop
WINDOW CLOSE 2
SCREEN CLOSE 1
END
DATA 180, 50, 210, 80, 210, 120
DATA 180,150, 100, 150, 70, 120
DATA  70, 80, 100, 50
```

```
SCREEN 1,320,250,2,1
WINDOW 2,,,0,1
LINE (180,50)-(210,80)
LINE (210,80)-(210,120)
LINE (210,120)-(180,150)
LINE (180,150)-(100,150)
LINE (100,150)-(70,120)
LINE (70,120)-(70,80)
LINE (70,80)-(100,50)
LINE (100,50)-(180,50)
loop:
IF MOUSE(0)=0 THEN loop
WINDOW CLOSE 2
SCREEN CLOSE 1
```

Poligoni
Ecco ben due listati Basic per disegnare poligoni. Il più breve non richiede l'uso di librerie; quello che, invece, richiede una digitazione più lunga, offre potenzialità decisamente migliori. Si legga la risposta a Sergio Giardina per ottenere tutte le delucidazioni del caso

Anche in questo caso i particolari non sono troppo chiari, ma l'inghippo è abbastanza facile da spiegare.

La tecnica cui accenna il lettore, ovvero la creazione di uno spazio "riservato" in memoria per depositarvi dei dati, o una struttura di dati, può anche essere affrontata adoperando allo scopo una variabile indicizzata, ma con le cautele del caso.

Questa tecnica va infatti adoperata solo se si è ben sicuri di come si muove un programma Basic, e se questo non è troppo complesso e fitto di subprogrammi o funzioni.

Più che la teoria, può essere esplicativo un esempio pratico. Si provi a digitare, dopo aver caricato Amigabasic, il seguente programma:

```
DEFINT a-z
DIM array(5000)
indir&=VARPTR(array(0))
PRINT "Indirizzo array =";
indir&
a = 555: b = 666: c =777:
d=6555
indir&=VARPTR(array(0))
PRINT "Indirizzo array =";
indir&
```

Molto banalmente, questo si limita a creare un array di 5000 elementi, ed a stamparne l'indirizzo di inizio ricavato tramite l'istruzione di Amigabasic **Varptr**.

Ciò fatto, vengono dichiarate alcune altre variabili a puro titolo dimostrativo, quindi viene di nuovo prelevato l'indirizzo di inizio dell'array e stampato sullo schermo.

Se si prova ad attivare il programma, si noterà come i due indirizzi visualizzati, sebbene siano riferiti allo stesso elemento dell'array, non saranno identici!

Il motivo è presto detto: successive assegnazioni di variabili, possono costringere (anzi, è quasi la norma) l'interprete Basic a spostare fisicamente l'intero l'array, rendendo così impossibile il rintracciamento dei corretti valori eventualmente pokati nelle locazioni riservate alla variabile indicizzata. Detto più crudamente, la struttura dati può anche andare a farsi benedire.

Se ci si vuole riservare un'area di memoria libera da interferenze, la soluzione più corretta resta dunque quella di ricorrere alla funzione **Allocmem** della libreria **Exec** (più volte descritta in questa rubrica, e della quale si riparlerà), oppure prestare una certa (notevole) attenzione allo sviluppo del programma Basic. L'inconveniente prima visto può infatti essere evitato con un banale accorgimento: dichiarare tutte le variabili all'inizio del programma, così come sarebbe obbligatorio in altri linguaggi come il **Pascal** o il **C**.

Si provi, a titolo di esempio, a modificare il listato precedente in modo che risulti così configurato:

```
DEFINT a-z
DIM array(5000)
a=0: b=0: c=0: d=0
indir&=VARPTR(array(0))
PRINT "Indirizzo array =";
indir&
```

a=555: b=666: c=777: d=6555
indir&=VARPTR(array(0))
PRINT "Indirizzo array =" ;indir&

Lanciato di nuovo il programma, stavolta i due indirizzi ottenuti risulteranno uguali. L'array sarà rimasto ben ancorato al suo posto grazie alla semplice dichiarazione iniziale delle variabili, per le quali l'interprete crea subito lo spazio necessario. Dovendolo fare in un momento successivo, si troverebbe invece costretto a far loro spazio, spostando quelle preesistenti (l'array). Volendo adottare questa tecnica, non va inoltre dimenticato che le variabili **Static** contenute in eventuali subprogrammi (**Sub...End Sub**) sono indipendenti dal corpo principale del programma, e quindi verrebbero in ogni caso "viste" solo al momento della loro prima assegnazione, a meno che non siano dichiarate **Shared**.

Per evitare complicazioni, non resta comunque che ribadire la maggiore affidabilità della funzione Allocmem, da preferire in ogni caso.

Basic e poligoni

Possibile che non esista un comando Basic per disegnare un poligono con più di 4 lati, come fa Line?
(Sergio Giardina)

Possibilissimo. D'altra parte, già con lo stesso comando **Line** è possibile riprodurre graficamente un poligono con un numero maggiore di lati, anche se con qualche riga in più di programma. Prima di prospettare una diversa soluzione, vediamolo subito con un breve listatino che si limita a creare uno schermo con relativa **finestra**, e disegnarvi un **ottagono**. Per abbandonare il programma, sarà sufficiente pressare il pulsante sinistro del mouse.

Come si può notare non è poi così complesso, e, volendo, si potrebbe anche stendere una funzione Basic adatta allo scopo.

Rivolgendosi, però, alle routine di sistema, un comando già bell'e pronto esiste, ed è anche piuttosto semplice da adoperare. Si sta parlando della funzione **Polydraw** della **Graphic Library**, accessibile con le consuete "manovre" anche da Basic.

Manovre che, in pratica, consistono nel rendere accessibile al programma il file **Graphics.bmap** presente nella directory Basicdemos del disco Extras. L'argomento è stato ormai trattato abbondantemente negli ultimi numeri della rivista, per cui limitiamoci stavolta a prendere in esame il listato di queste pagine, che utilizza anche la funzione **Move** sempre della libreria grafica.

Ricordiamo solo che, a meno di non copiare il file Graphics.bmap nella directory corrente del Basic, si può modificare l'istruzione Library in modo che il suo argomento diventi qualcosa come Library **Df1:graphics.library**, con **df1:** da sostituire eventualmente con il nome della directory ove è rintracciabile il file **.bmap**.

Se lo si copia e lo si manda in esecuzione, questo si limiterà, come il precedente, a disegnare un ottagono, con due stringhe in mezzo.

Si noti, tra l'altro, come utilizzando le librerie, e non i soliti comandi di Amigabasic, sia possibile allineare una stringa dove più aggrada, ovvero non obbligatoriamente alla tradizionali "posizioni carattere". La stringa sulla riga inferiore, infatti, risulterà sfalsata rispetto a quella superiore, anche nella posizione dei singoli caratteri. Ciò è dovuto all'utilizzo di Move, la cui sintassi di chiamata è...

Move& (Rasterport&, x, y)

...con **x** ed **y** che rappresentano le coordinate orizzontali e verticali espresse in pixel, mentre **RasterPort** corrisponde alla funzione Basic **window(8)**. In pratica si ottiene lo stesso effetto del comando Basic **Locate** (si veda il manuale), ma con la possibilità di collocarsi in qualunque punto (pixel) dello schermo, mentre Locate è limitato alle posizioni-carattere.

Ma il fulcro della routine è Polydraw. Questa funzione consente di specificare una serie di coordinate all'interno di una variabile indicizzata (array), che verranno automaticamente unite tra di loro da linee. Vediamone subito la sintassi:

```
Polidraw&(Rast&,X,indir&)
```

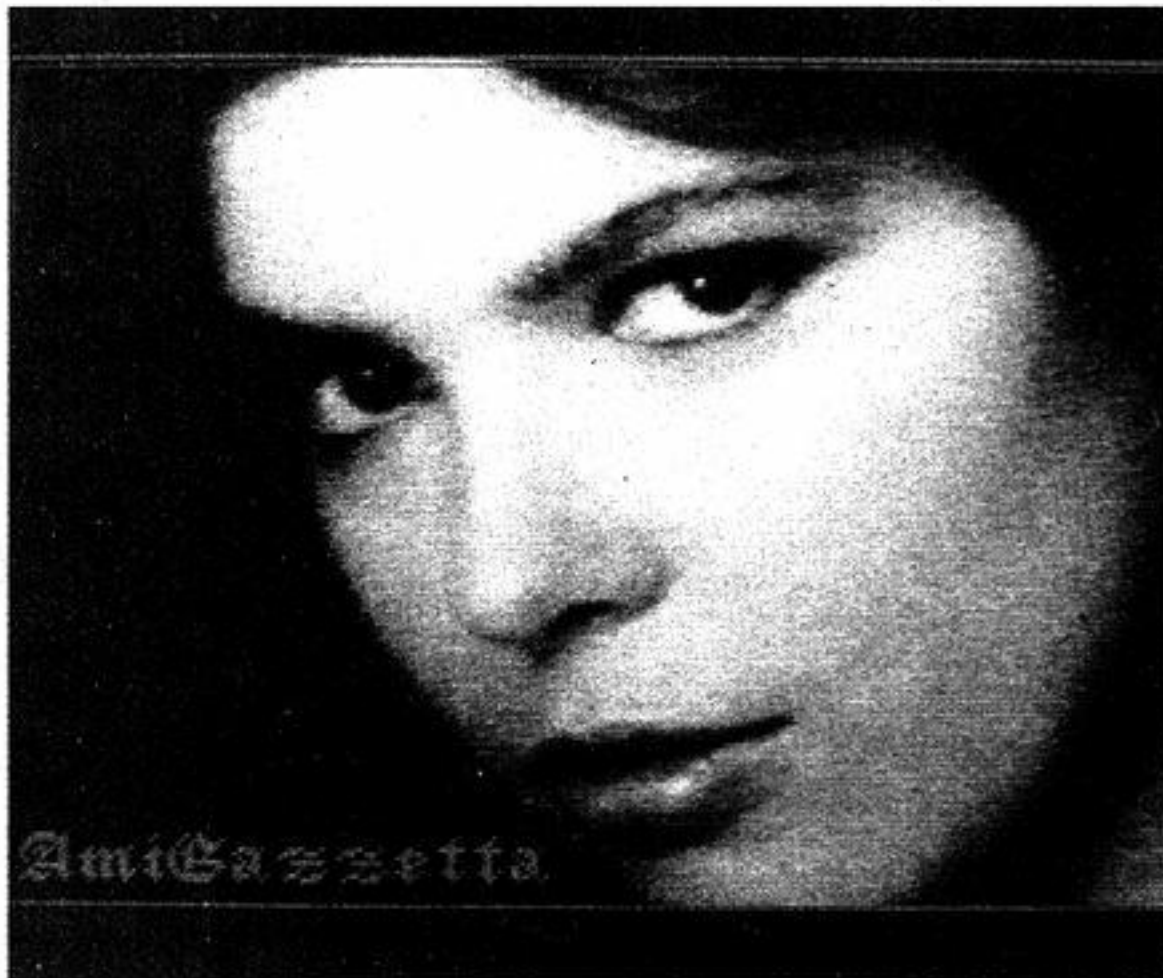
Il parametro **Rast&** è sempre l'indirizzo della Raster Port, ovvero window(8).

Indir& specifica invece l'indirizzo di inizio della variabile contenente le coordinate che interessano. Trattandosi di un array, l'inizio corrisponderà al suo elemento **0** (si veda il listato), ottenibile tramite il comando Basic **Varptr**.

X, infine, specifica il numero di coppie di coordinate da prendere in considerazione per il disegno. Attenzione al fatto che si è detto *"coppie"*, e non *"singoli valori"*, per cui questo numero corrisponderà in pratica al numero dei **lati** del poligono, ed alla metà degli elementi dell'array.

Come si può notare dal listato, basta quindi adoperare un ciclo for...next per inserire le coordinate (da considerare a gruppi di due, una verticale ed una orizzontale) in un array (**a%** nel listato), ricavarne l'indirizzo di inizio con Varptr, e passare il tutto alla funzione Polydraw.

Più semplice a farsi che a dirsi, la difficoltà maggiore consiste probabilmente nel calcolare esattamente le coordinate che saranno congiunte da Polydraw, ma in fondo si può anche procedere per tentativi...

Non resta da aggiungere che, per quanto questa funzione supplisca ad una deficienza del Basic, diventa molto più utile adoperando linguaggi meno ricchi (per non dire privi) di istruzioni di alto livello, come per esempio l'Assembly.

Comandi già pubblicati

A partire dal n.75 della rivista, si è cominciato ad esaminare in profondità il **Dos 1.3** di Amiga, rivolgendosi soprattutto (ma non solo) a chi si avvicina per la prima volta a questo computer. Chiaro che, con il succedersi degli appuntamenti, l'integrazione tra quanto trattato in precedenza e gli argomenti affrontati in queste pagine si fa sempre più stretta, anche se viene volutamente mantenuta la maggiore autonomia possibile. Per quei lettori che inizino solo ora a seguire la nostra rivista, è consigliabile (anche se non indispensabile) procurarsi i numeri a partire dal già citato **75**: si disporrà così di un completo compendio da affiancare alla consultazione del manuale, spesso fin troppo succinto nelle sue indicazioni. Ecco comunque un elenco di quanto già sviscerato nelle pagine della rubrica **Amigafacile**.

n.75
assign
copy
date
dir
install
path
search
sort

n.76
car.spec.
delete
format
protect
rename

n.77
execute
batch
if
skip...lab
quit

n.78
cd
ed
break

n.79
which
newshell
newcli
ser: par: e prt:
con: e newcon:
nil: e raw:

n.80
setmap
iconx
list

n.81
avail
alias
join

FAILAT

Si tratta di un comando espressamente dedicato alla gestione interna dei **Batch File**, funzionale soprattutto alla intercettazione di eventuali errori verificatisi nel corso dell'esecuzione.
Tutti i comandi del Dos ,inseriti in una sequenza batch, concludono in pratica la loro attività restituendo il controllo al file **script** unitamente ad un codice di errore. In genere, se tutto si è svolto senza problemi, questo codice sarà uguale a zero, in caso contrario il valore-codice risulterà tanto più alto quanto più "grave" sarà l'errore nel quale si è incorsi.
Se un errore interviene durante l'esecuzione di un batch file, questo viene di norma interrotto, con un ritorno all'ambiente Shell (o Cli) accompagnato da adeguata segnalazione. Utile per rilevare inconvenienti di programmazione, in molti casi è tuttavia auspicabile un controllo diretto sugli errori, in modo da poterli gestire autonomamente o addirittura sfruttarli per particolari implementazioni. Ed ecco che entra in ballo Failat.

Questo comando imposta infatti una "soglia di attenzione" per i codici di errore, al di sotto della quale essi non provocano l'interruzione di un batch file, pur segnalando (in alcuni casi) a video l'eventuale enpasse verificatasi.

Failat

Impartito senza alcun parametro, segnala l'attuale stato della soglia d'errore, stampandone sullo schermo il valore del codice. Per default questo valore corrisponde a **10**, a meno ovviamente che non venga modificato dalla **Startup-Sequence** del disco adoperato per il boot di Amiga.

Codice

E' il valore che si vuole impostare come nuova soglia di errore. Nella maggior parte dei casi, per consentire una intercettazione degli errori, è sufficiente immettere un valore superiore a **20**, ma nulla vieta che si adoperi un parametro **100**, o anche superiore.
Il valore 20 come limite ha senso in quanto, come forse noto, la condizione **If** consente di "scremare" il livello di errore mediante le parole chiave **Warn**, **Error** e **Fail**, che nell'ordine vagliano i "return-code" con valore uguale o maggiore a **5**, uguale o maggiore a **10**, ed infine uguale o maggiore a **20** (si veda anche la descrizione di If sul **n. 77**).

Failat, approfondimenti ed esperimenti

Per capirne concretamente il funzionamento, si provi questo semplice esperimento, che potrebbe anche tornare utile in alcune circostanze. Anzitutto, dopo aver attivato il sistema con il disco **Workbench**, si apra una finestra **Shell** e al suo interno si digiti:
Mount Rad:.
Così facendo, si provoca l'installazione della cosiddetta **Recoverable Ram**, ovvero una Ram Disk resistente al reset (e al Guru, anche se non proprio sempre). Senza dilungarci sulla **Rad**, argomento che avremo occasione di riaffrontare in altra sede, limitiamoci per ora ad impartire un comando **Info** per constatare se essa sia effettivamente presente nella lista delle periferiche attive. In caso contrario, si provi eventualmente a "citarla" con un banale **List Rad:**, e quindi a ripetere **Info**.
Una volta certi della sua effettiva installazione, entriamo nel vivo dell'esperimento per mettere alla prova **Failat**. Si ripeta, dunque, il comando Mount Rad: (con i due punti!). Si otterrà in risposta una segnalazione...
Device rad: already mounted
...che sta più o meno per *"è già montata"*.
Ora facciamo eseguire lo stesso compito ad un batch file. Visto che abbiamo a disposizione la Rad, "portiamoci" al suo interno con **Cd Rad:**, ed editiamo un mini file-script di nome **Test** adoperando **Ed** o una delle tecniche cui abbiamo più volte fatto cenno; per esempio:
Copy * to rad: test
...e poi...
**Ctrl + **
...per concludere l'editing. Il file dovrà contenere queste sole due righe:

Mount Rad:
Echo "Tutto fatto"

Ora lo si lanci impartendo **Execute Test**. Si otterrà la stessa segnalazione di errore prima vista, seguita da "**mount failed returncode 10**". La seconda riga di istruzioni, in pratica, non verrà mai eseguita, in quanto lo script si interrompe prima di giungervi.

Ne abbiamo però ricavato il codice di errore, anche se non sarebbe poi così importante. Si provi ora a editare un altro file, che chiameremo **Test2**, così configurato:

Failat 20
Mount Rad:
If error
Echo "E' già montata!"
Endif
List Rad:
Failat 10

Lanciato con **Execute test2** (oppure con solo Test2 se si provvede a settare il bit "script" con il comando **Protect**, descritto su Amigafacile n. 76), stavolta si otterrà solo la "nostra" segnalazione che la Rad è già montata, e nonostante l'errore del Dos, verrà ugualmente eseguita l'istruzione List posta in coda allo script. In altre parole, l'esecuzione del batch file non verrà interrotta, grazie al **Failat 20** che introduce la sequenza di comandi.
Una volta innalzata la soglia di attenzione, l'errore con codice 10, provocato dal tentativo di reinstallare la Rad, non sarà più fatale, mentre risulterà invece intercettabile da **If Error**, consentendo l'esecuzione di quanto inserito all'interno della condizione **If... Endif**, ma anche di proseguire poi con altre istruzioni. Non va dimenticato, una volta terminate le operazioni che interessano, il ripristino della normale soglia d'errore, ovvero un esplicito **Failat 10** (a meno di esigenze particolari, s'intende).

Un esempio del genere potrebbe tornare utile se si pensa di utilizzare spesso la Rad (senza renderla autobootante), e quindi si intende inserirne il comando di installazione nella startup - sequence del proprio disco di lavoro. Ad ogni nuovo boot, senza un accorgimento del genere, la procedura di start si interromperebbe ogni volta.
Ci si ricordi, ultimato l'esperimento, che per eliminare fisicamente la Rad è necessario impartire **RemRad**, oppure spegnere il computer.

Nota bene

Le parole chiave di ogni comando sono rappresentate in maiuscolo e vanno (eventualmente) adoperate così come sono. In minuscolo sono invece riprodotti i parametri che vanno ridefiniti dall'utente.

EVAL

Questo comando consente di disporre di una piccola calcolatrice sempre a portata di mano quando si lavora in ambiente **Shell**. In verità non è il caso di parlare di una calcolatrice vera e propria, soprattutto se si pensa alla miriade di programmi del genere (per lo più di pubblico dominio) che emulano le più sofisticate aparechiature in commercio, comunque può risultare utile in più di un'occasione.
Il limite più evidente è rappresentato dalla obbligatorietà di utilizzare numeri interi, per cui non sono possibili calcoli di estrema precisione.
D'altra parte, soprattutto nella programmazione di "basso livello", spesso si ha la necessità di effettuare calcoli del genere, che anche vadano oltre le semplici 4 operazioni aritmetiche, ed in questo senso Eval è dotato di sufficiente versatilità.

E' il simbolo dell'operazione da eseguire sui due termini numerici. Importante da ricordare che è indispensabile uno **spazio** prima e dopo il simbolo qui inserito. Sono ammessi i comuni operatori aritmetici (**+**, **-**, *****, **/**), ma anche **Mod** (per chi non lo sapesse, indica il **resto** di una divisione effettuata tra il primo ed il secondo operando), nonché tutta la gamma delle operazioni **logiche** o di **scorrimento** di bit che servono soprattutto a chi non disdegna la programmazione. Ecco una tabella di questi ultimi, con il simbolo da apporre indicato nella prima colonna:

<table>
<tr><td>mod</td><td>modulo</td><td>xor</td><td>Xor</td></tr>
<tr><td>&</td><td>And</td><td>eqv</td><td>Eqv</td></tr>
<tr><td>|</td><td>Or</td><td><</td><td>shift sin.</td></tr>
<tr><td>~</td><td>Not</td><td>></td><td>shift des.</td></tr>
</table>

Si badi che, nel caso di **Not**, la forma da adottare diverrà qualcosa come **Eval ~4**, ovvero con il simbolo ed un unico valore, mentre negli scorrimenti il primo parametro sarà il numero da trattare, e il secondo "num" indicherà di quanti bit deve essere fatto scorrere. Quindi, per es., con Eval **1 < 7** si otterrà **128**, in accordo alla logica della notazione binaria.

Per il secondo termine (num) dell'operazione, per il quale valgono le stesse considerazioni fatte sul primo, va aggiunto che Eval consente operazioni "miste", ovvero con i due termini espressi in differenti notazioni. Si potrà quindi adoperare...
Eval 0xff * 10
... che darà come risultato 2550 (ff esadecimale = 255), o anche...
Eval 0xff * 0x10
...stavolta con entrambi i valori in esadecimale, con risultato 4080. A meno che non si adoperi l'ultimo parametro (Lformat), l'esito dell'operazione viene comunque espresso in notazione decimale.
Con TO seguito dal nome di un file, eventualmente associato al suo percorso, si ha una normale redirezione, ovvero il risultato dell'operazione verrà trascritto su quel file. Lo stesso risultato si può ottenere anche con una tradizionale redirezione secondo i dettami del Dos di Amiga, ovvero inserendo >nomefile subito dopo Eval e prima dell'inizio dei parametri.

EVAL num simbolo num TO LFORMAT="stringa"

Num. E' il primo termine dell'operazione. Il numero deve essere di tipo intero, comunque Eval non rifiuta l'eventuale immissione di un valore **frazionario**, si limita semplicemente ad ignorarlo, effettuando ugualmente l'operazione sulla sola parte intera. Si può specificare questo parametro anche da solo, senza farlo seguire dal secondo termine dell'operazione; in questo caso si avrà in output lo stesso numero immesso subito dopo Eval. Apparentemente inutile, questa caratteristica si presta invece ad un'applicazione immediata: la conversione da un numero **esadecimale** oppure **ottale**. Eval (e questo è davvero comodo) può infatti prendere in considerazione tre notazioni numeriche: **decimale**, esadecimale, ed ottale. Per default, viene assunta la nostra cara notazione decimale, adottata se un numero non è dotato di alcun prefisso. Per indicare un valore esadecimale occorre invece premettere **0x** (oppure **#x**), mentre un solo **#** (oppure un solo **0**, da evitare per non creare confusione) indicherà la notazione ottale.

Tornando al nostro parametro, se lo si inserisce sotto forma di valore (p. es.) esadecimale, senza farlo seguire da altro, dopo il return se ne otterrà, in pratica, la sua conversione in decimale. Si provi per esempio a digitare **Eval 0xff** (seguito da Return): sullo schermo si vedrà comparire **255**, ovvero la sua conversione in decimale (si veda anche il riquadro Esempi).
La stessa procedura può essere infine applicata ai caratteri. Il parametro **"num"** può infatti essere sostituito da un carattere, purchè preceduto dal simbolo apostrofo ('): Eval ne prenderà in considerazione il suo codice **Ascii**. Anche in questo caso, se si utilizza num da solo, si dispone di un modo rapido per conoscere l'equivalente Ascii di un simbolo: con **Eval 'A** si avrà dunque in risposta **65**, e così via.
Come ovvio, lo stesso principio può essere adottato nell'eventualità di un'operazione vera e propria nella quale uno dei due termini debba essere il codice Ascii di un carattere.

Lformat. Aggiungendo alla sintassi di **Eval** il parametro Lformat, è possibile modificare l'**output** del comando. E questo non solo a fini estetici, ma anche sostanziali. Si può, infatti, aggiungere una **stringa** che verrà riprodotta assieme all'output, ma in ogni caso specificando il formato in cui verrà espresso il risultato. Per far ciò, vanno adoperati alcuni simboli, e più precisamente: *percento / numero* (**%X**) seguito da un numero, che forzerà il risultato ad essere espresso in notazione esadecimale, mentre il numero indicherà quante cifre utilizzare (se in sovrappiù, saranno inseriti automaticamente degli zeri all'inizio). Con **%N** si avrà invece il risultato espresso in decimale, senza la necessità di specificare alcun numero. Per un risultato espresso sotto forma di carattere, basterà poi utilizzare **%C**.
Ai fini della formattazione vera e propria, è anche adoperabile ***N** (da **non** confondersi con **%N**), per ottenere in pratica un ritorno carrello (un *a capo*, insomma) nel punto in cui è inserito tale simbolo. Utilizzando al meglio questo parametro, è possibile quindi sfruttare Eval come calcolatrice, ma con traduzione nella notazione che più aggrada. Basta ricorrere, come ovvio, a dei batch file opportunamente configurati come quelli mostrati nel riquadro a parte, utili soprattutto per chiarire quanto appena espresso.

Nome del batch = ASC
Sintassi = (execute) ASC carattere
Azione = Fornisce il codice Ascii di "carattere"
Esempio d'uso = *Asc A* (risultato: 65, se A in maiuscolo)

```
.key car
 if x<car> eq "x"
  echo "Sintassi:"
  echo "ASC carattere"
  quit
 endif
eval '<car>
```

Nome del batch = CHAR
Sintassi = (execute) CHAR cod
Azione = Mostra il carattere associato al codice ascii "cod"
Esempio d'uso = Char 65 (risultato: A)

```
.key num
 if x<num> eq "x"
  echo "Sintassi:"
  echo "CHAR codiceascii"
  quit
 endif
eval <num> lformat="*N%C*N*N"
```

Nome del batch = ESA_DEC
Sintassi = (execute) ESA_DEC valore
Azione = Converte l'esadecimale "valore" in decimale
Esempio d'uso = Esa_dec ffff (risultato: 65535)

```
.key num
 if x<num> eq "x"
  echo "Sintassi:"
  echo "Esa_Dec numesadecimale"
  quit
 endif
 eval 0X<num>
```

Nome del batch = DEC_ESA
Sintassi = (execute) DEC_ESA numero
Azione = Converte il decimale "numero" in esadecimale
Esempio d'uso = Dec_esa 64533 (risultato: 0000FC15)

```
.key num
 if x<num> eq "x"
  echo "Sintassi:"
  echo "DEC_ESA numdecimale"
  quit
 endif
eval <num> lformat="*N%X8*N*N"
```

Eval in pratica

Per applicare quanto descritto a proposito di **Eval**, niente di meglio che realizzare quattro brevi e semplici nuovi "comandi", dedicati tutti alla conversione tra diverse notazioni. In questi file non vengono effettuati veri e propri calcoli, peraltro molto semplici da sperimentare in proprio, comunque ci si può trovare tutto il necessario per capire il funzionamento di Eval, soprattutto in relazione al paranetro **Lformat**.

Si consiglia, pertanto, di copiare i **Batch** con l'editor che si preferisce e, dopo averli salvati (il nome da noi assegnato è ovviamente facoltativo), impartire un comando...
Protect nomebatch +S
... in modo da non dover adoperare **Execute** per mandarli in esecuzione: basterà digitare il loro nome (con l'eventuale percorso).
Nella descrizione che compare negi quattro riquadri, Execute è usato tra parentesi, proprio per indicare questa possibilità. Qualora non si adoperasse Protect + S, andrebbe sempre inserito Execute prima del nome del batch.
Perché i file funzionino, nella directory **C** di sistema dovranno obbligatoriamente essere presenti i comandi **Eval, Echo, If, Endif** e **Quit**.

di Luigi Callegari

AMIGA, INIZIARE CON IL "C"

Iniziamo, finalmente, a parlare del linguaggio compilatore ideale per un computer sofisticato quale è l'Amiga

Spesso chi si avvicina al linguaggio **C** ha la terribile sensazione che sia completamente differente dal caro vecchio **B**asic. In effetti, però, molti dei concetti cardine appresi in quest'ultimo linguaggio di programmazione sono riutilizzabili anche in C, sebbene sia necessario imparare a pensare i propri problemi software in termini diversi, in modo da scrivere programmi che sfruttino efficientemente la potenza del linguaggio.

Una volta appresa la sintassi delle **istruzioni** e dei **costrutti** del C, la maggior parte della difficoltà di apprendimento deriva soltanto dal dover utilizzare un **compilatore** invece che un **interprete**.

Dal momento che i compilatori per Amiga sono ormai standardizzati secondo i dettami **ANSI**, tutto ciò che concerne la struttura di base del linguaggio, e la stesura di semplici routines, è comune a quanto già detto recentemente e contestualmente da Giancarlo Mariani sulle pagine di CCC per quanto riguarda i compilatori Ms - Dos.

In ogni caso, per consentire una familiarizzazione rapida e, speriamo, interessante ai possessori di Amiga, daremo un taglio molto pratico alle nostre chiacchierate sulla programmazione in linguaggio C, che da ora presenteremo mensilmente. Questa volta illustreremo l'utilizzo dei **vettori** di valori interi e delle **funzioni** basilari per la gestione di files e caratteri **ASCII**.

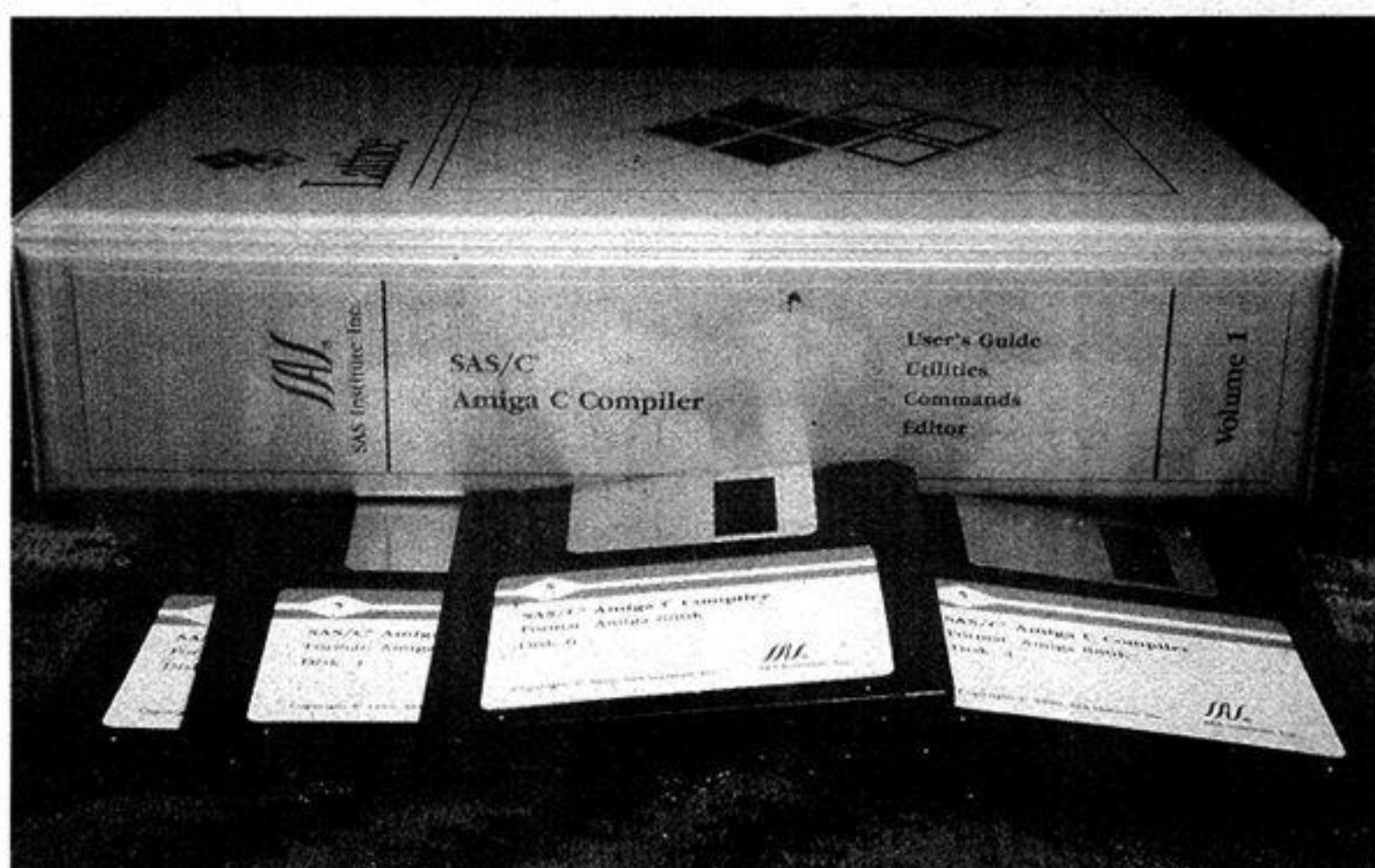
Input / Output

Vi sono due insiemi di funzioni in linguaggio C che constentono di accedere a **files** ed a **devices**: le cosiddette funzioni di I/O "**senza** buffer" e le funzioni di I/O "**con** buffer".

Nella documentazione di Amiga si fa riferimento spesso a questi due insiemi chiamandoli, rispettivamente, funzioni di "livello 1" e funzioni di "livello 2". Le funzioni standard, "bufferizzate", usate dai compilatori C di Amiga (**Lattice** e **Aztec**) sono compatibili con lo standard ANSI C, mentre quelle non bufferizzate possono funzionare in maniera leggermente diversa a seconda del compilatore, se non nella sostanza, quantomeno nella tecnica di implementazione da parte dei realizzatori del compilatore.

In effetti, i compilatori C di Amiga consentono anche di utilizzare funzioni del cosiddetto **Livello 0**, ovvero le chiamate dirette alle routines della **libreria** AmigaDOS del sistema operativo di Amiga, che sono spesso molto simili (anche per nome e funzione svolta) alle funzioni di base UNIX od ANSI.

Ovviamente, i programmi che utilizzeranno funzioni di livello 1 e 2 saranno facilmente trasportabili, ovverossia compilabili anche sotto sistemi operativi di-

versi (UNIX, Ms - Dos) e con differenti computer, senza problemi, purchè ovviamente si utilizzino compilatori ANSI C. Invece programmi utilizzanti funzioni di livello 0 gireranno **solo** con Amiga, dal momento che soltanto il suo sistema operativo prevede certe funzioni specifiche in ROM.

Tra l'altro, spesso le funzioni si chiamano in modo molto simili a livello 0 e 1 e svolgono compiti simili o adddirittura eguali. Quello che cambia è, ovviamente, il tipo di codice generato dal compilatore e i parametri, da passare e restituiti dalla funzione.

Funzioni di I/O

Le funzioni di I/O **non** bufferizzate sono così chiamate perchè, con alcune eccezioni, trasferiscono immediatamente i dati dal programma al file o device.

Le funzioni **bufferizzate**, invece, utilizzano un'area di memoria, detto in gergo "buffer", durante i trasferimenti tra le aree di memoria usate dal programma ed il file o device.

Da ciò si deduce che le funzioni di I/O non bufferizzate sono utilizzate dai programmi che eseguono da sè il raggruppamento in blocchi, e la suddivisione degli stessi durante le riletture, mentre le funzioni standard sono più adatte a programmi dove non è particolarmente importante il potersi strutturare da sè i blocchi di dati trasferiti.

In generale, un buon programmatore tenderà ad usare le funzioni di I/O non bufferizzate, che consentono di aumentare l'efficienza del programma (se non addirittura le funzioni di chiamata diretta delle routine di AmigaDOS, qualora il programma non debba risultare trasportabile sotto altri computer) al costo di qualche piccolo sforzo di programmazione in più. I **principianti**, o coloro i quali vogliono scrivere routines velocemente, senza doversi troppo concentrare sulla gestione dei files, useranno le funzioni di livello più alto.

La sequenza di operazioni - tipo per l'utilizzo di un file è sempre eguale per tutti i livelli: il file deve essere aperto (**open**), ovvero si deve avvertire il sistema operativo che si vuole eseguire una serie di operazioni con esso; poi si utilizzano le apposite funzioni di posizionamento (**seek**), lettura (**read**) o scrittura (**write**) e, alla fine, il file può essere chiuso (**close**), ovvero si può indicare al Sistema Operativo (S.O.) che non è più necessario utilizzarlo e che quindi possono essere liberate le aree di memoria dedicate alla sua gestione, restituendolo all'accesso libero da parte di altri programmi.

Ciascun livello ha le sue proprie funzioni di apertura e chiusura dei files, come riportato in figura 1. Vediamo quindi le differenze di utilizzo delle funzioni di gestione files più in dettaglio.

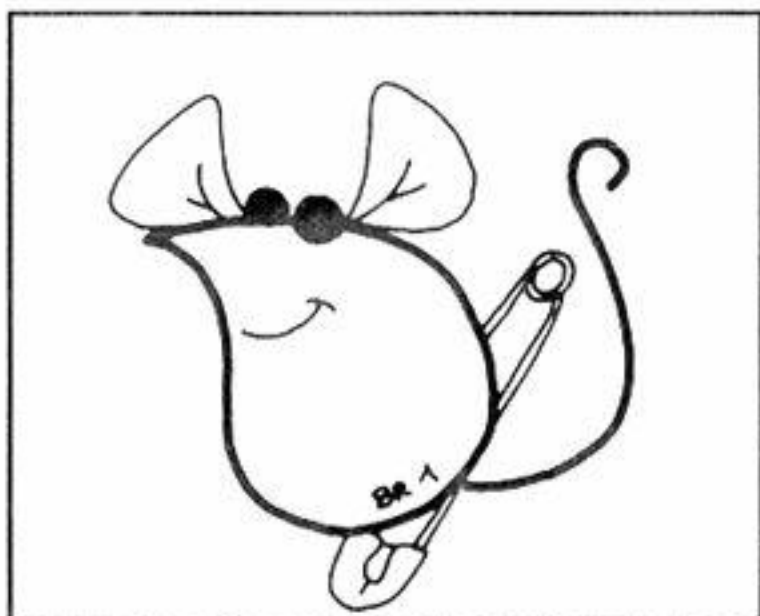
Aprire a livello 2

Per aprire un file a livello 2 occorre decidere il nome del file ed il modo con il quale lo si vuole aprire. Il nome del file è una **stringa alfanumerica** che rappresenta il nome del file secondo gli standard AmigaDOS, ovvero con il doppio punto (:) che indica il termine di un nome di device fisico o logico (**PRT:**, **DH0:**, **DF1:**, **LIBS**: eccetera) e lo slash (/) che

```
 1  #include <ctype.h>
 2  #include <stdio.h>
 3
 4  #define TRUE 1
 5
 6  int count[ 26 ];
 7
 8  void die( char *s )
 9  {
10    printf( "%s\n", s );
11    exit( 0 );
12  }
13
14  void main( void )
15  {
16    int c = 0, nochar = 0, d;
17    char nomefile[ 30 ];
18    FILE *handle;
19
20    printf( "Nome del file ?" );
21    scanf( "%s", &nomefile[0] );
22
23    handle = fopen( nomefile, "r" );
24    if ( handle == NULL )
25      {
26         die( "File inapribile!" );
27      }
28
29 while ( ( c = getc( handle )) != EOF)
30      {
31         c = toupper( c );
32
33         if ( isalpha( c ) == TRUE )
34            count[ c - 'A' ] += 1;
35         else
36            ++nochar;
37      }
38
39 for (c=0, d='A'; c < 26; c+=2, d+=2 )
40      {
41      printf( "  %c=%6d ********
%c=%6d\n",\
42          d, count[c], d+1, count[c+1]);
43      }
44
45    printf("Chars non alfabetici:
%d\n\n", nochar );
46  }
```

Il micro listato in C
Al contrario dell'articolo, che bisogna leggere un paio di volte per comprenderlo a fondo, il programma dimostrativo è di una semplità tale da esser consigliato anche ai principianti

indica il termine di un nome di directory prima dell'inizio del nome del file o della subdirectory successiva. La stringa deve ovviamente terminare col codice di NULL ('\0') come vogliono le convenzioni del C.

Il modo di apertura è una stringa alfanumerica che indica il tipo di accesso desiderato sul file: "**r**" indica lettura, "**w**" indica scrittura, "**a**" indica aggiunta in coda ai dati esistenti. Le seguenti linee in un programma, ad esempio, sono ritenute valide:

```
#include <stdio.h>
.. .. .. ..
void dummy(void)
{
  FILE *fp;
  fp = fopen( "df1:Avv/Cristina",
"w" );
.. .. .. ..
}
```

...nel qual caso si otterrà l'apertura del file chiamato *"Cristina"* presente nella directory *"Avv"* del dischetto inserito nel primo drive esterno (od il secondo interno negli A-2000). Tale file sarà usato per scrivere dati, come indica la *"w"*. Ovviamente i parametri avrebbero potuto essere passati come puntatori a stringhe contenenti i dati...

```
void dummy(void)
{
  FILE *fp;
  char s[ 50 ];
  printf( "Nome file?" );
  gets( s );
  fp=fopen( s, "r" );
 .. .. ..
}
```

...aprirà il file il cui nome è stato inserito da tastiera, trasferito dalla fonte di input standard (cioè la tastiera del computer, di norma) al vettore di caratteri "**s**" ampio **50** bytes, per la sola lettura.

La funzione **fopen()** che apre un file restituisce un valore di tipo FILE. Tale tipo è definito nel file di inclusione "**stdio.h**", per cui è necessario includerlo con l'apposita **#include** nel listato. Tale parametro di tipo FILE, che materialmente è un puntatore ad una struttura dati gestita internamente dal compilatore, chiamato generalmente "filehandle" o "handle" ed utilizzato dalle altre funzioni di livello 2 (di lettura, scrittura, posizionamento eccetera) per gestire il file stesso.

Apertura di un file
2) FILE *fp = fopen(char *nome, char *modo)
1) int fh = open(char *nome, int modo, int prot)

Chiusura di un file
2) int errore = fclose(FILE *fp)
1) int errore = close(int fh)

Posizionamento in un file
2) int errore = fseek(FILE *fp, long rpos, int modo)
1) long apos = lseek(int fh, long rpos, int modo)

Lettura da un file
2) int conteggio = fread(char *b, int bsize, int max, FILE *fp)
1) unsigned cont = read(int fh, char *buffer, unsigned lun)

Scrittura in un file
2) int conteggio = fwrite(char *b, int bsize, int max, FILE *fp)
1) unsigned cont = write(int fh, char buffer, unsigned lun)

Fig. 1. Funzioni generiche di I/O a livello 1 e 2

In uno stesso programma si possono aprire più files, fino a **20** tipicamente, ma ciascuno sarà dotato di un filehandle univoco.

Il valore restituito da fopen() è pari a **NULL (0)** se l'apertura non è stata possibile, per cui il programma può eseguire un controllo e prendere provvedimenti in caso di errore.

Aprire a livello 1

L'apertura di un file a Livello 1 è molto simile alla precedente. Occorre avere un nome, come prima, ed un modo di accesso. La differenza è che **Lattice C** consente un **terzo** parametro in un caso particolare di modo di apertura per compatibilità UNIX, effettivamente ignorato, mentre **Aztec C** non lo prevede.

I modi di apertura sono indicati a Livello 1 non con una lettera ma con una **stringa maiuscola**, definiti nel file di inclusione "**fcntl.h**".

O_RDONLY (Solo lettura)
O_WRONLY (Solo scrittura)
O_RDWR (Lettura e scrittura)
O_APPEND (Posiziona a fine file)
O_TRUNC (Svuota il file)
O_CREAT (Creazione del file)
O_EXCL (Non ricreare file)

Si noti che le costanti fondamentali sono le prime tre, mentre le altre sono attributi. O_APPEND viene usato con la seconda o la terza per indicare che tutte le scritture nel file devono avvenire alla coda dello stesso (quindi aggiungendo dati a quelli pre-esistenti). O_TRUNC è usato con O_CREAT, O_WRONLY o, O_RDWR ad indicare che se il file esite deve essere troncato a lunghezza zero. O_CREAT indica che se il file non esiste deve essere creato e O_EXCL è usato con O_CREAT ad indicare che se il file esiste già la funzione **open()** deve restituire un errore.

Vediamo un esempio di linee-tipo in un programma più complesso:

```
#include <fcntl.h>
void dummy(void)
{
int fileh;
fileh = open( "ram:Monica",O_RDWR )
 .. .. ..
}
```

```
Lattice Screen Editor
{
    printf("\nErrore %d: %s\n",n,s);
    exit( (LONG)EX_ERROR );
}

/* Alloca memoria per un nuovo nodo */
struct nodo *palloc( void )
{
    return( (struct nodo *) malloc(sizeof(struct nodo)));
}

/* Alloca memoria e salva una parola nuova */
char *savestr( char *s )
{
    register char *p;

    p = (char *) malloc( strlen( s ));
    if ( p == NULL ) return( p );
    else             return( (char *)strcpy( p, s) );
}

/* Stampa ricorsivamente l'albero */
void displyalbero( struct nodo *p )
{
    if ( p != NULL )
    {
        displyalbero( p -> min );
LINE: 81    FILE: Vtext.c                    INSERT (1)
Text entry mode. Press F1 for LSE help.
```

...apre il file chiamato *"Monica"* presente nel RAM DISK per lettura e scrittura contemporanea. Il numero intero rappresentante il filehandle di livello 1 (da non confondere assolutamente con il filehandle di tipo (FILE *) restituito dal fopen() di livello 1) viene memorizzato nella variabile **fh**. Il valore è pari a **-1** solo se l'apertura del file non ha avuto successo.

Lettura

Il concetto della lettura dei dati da un file si basa, ad ambedue i livelli, sul trasferimento dei dati contenuti nel file, di cui si è ricavato un handle tramite l'apposita funzione di apertura, in un buffer di memoria vista dal programma come una sequenza di dati di tipo carattere (**char**).

La funzione di livello 2 **fread()** usa leggere dei blocchi di dati. Il **primo** parametro è un puntatore al buffer allocato per conservare i dati letti (solitamente una stringa di caratteri); il **secondo** è la dimensione di ciascun blocco in bytes; il **terzo** è il numero di blocchi da leggere e l'**ultimo** parametro è l'handle del file aperto con fopen().

La funzione restituisce un numero intero corrispondente al numero di blocchi effettivamente letto, che in caso di lettura senza problemi deve essere eguale al *terzo* parametro, altrimenti significa che si è verificato un errore di lettura o che si è incontrata la fine del file stesso.

Facciamo un esempio:

```
#include <stdio.h>
void dummy( void )
{
  int a;
  FILE *fp;
  char bloc[ 512 ];
  fp = fopen( "Teresa", "r" );
  if ( fp == NULL ) exit( 5 );
  a = fread( bloc, 1, 512, fp );
  if ( a != 512 ) {
     printf( "Lettura errata\n");
  .. .. .. ..
  Close( fp );
}
```

...eseguirà l'apertura del file *"Teresa"* nella directory corrente per sola lettura, verificando che l'operazione sia riuscita altrimenti esce dal programma con **exit()**, che restituisce un valore di **5** al processo CLI che ha invocato il programma. Poi si prosegue leggendo un blocco da **512** bytes dal file indicato dall'handle **fp** nello spazio indicato dalla variabile **bloc**, appositamente dimensionata per contenere blocchi di questo tipo. Nel caso la lettura non sia corretta, si genera con **printf()** un messaggio di errore. La funzione **Close()** posta alla fine chiude il file precedentemente aperto, funzione peraltro svolta automaticametne dal codice di inizializzazione inserito dal linker del linguaggio C, quando il programma termina o si usa exit().

A livello 1 le cose si svolgono in modo analogo, incentrando la lettura **non** su blocchi di bytes, bensì sui **singoli** bytes. Il primo parametro della funzione open() è il filehandle restituito da un precedente open (per il file desiderato); il secondo parametro è il puntatore al buffer predisposto per la lettura: il terzo indica il numero di caratteri che si desidera leggere. La funzione restituisce un intero senza segno (**unsigned**) corrispondente al numero di caratteri effettivamente letti senza problemi, quindi se non combacia con l'ultimo parametro di open() significa che si è incappati in un errore oppure nella fine del file. Vedremo in futuro come si possono distinguere da programma queste due possibilità. per ora accontentiamoci di un esempio semplice:

```
#include <fcntl.h>
void main( void )
{
   int fh;
   unsigned tot;
   char s[500];
   fh = open( "Mariella", O_RDONLY
);
   if ( fh == -1 ) exit( 5 );
   tot=read( fh, s, 500 );
   if ( tot != 500 ) {
      printf( "Errata lettura!\n");
  .. .. ..
}
```

...qui si apre un file a Livello 1 chiamato *"Mariella"*, nella directory corrente, per sola lettura, uscendo in caso di fallimento del tentativo di open(). Poi si leggono,

Lettura di un carattere da file
int c = fgetc(FILE *fp)
int c = getc(FILE *fp)
int c = getchar()
int c = getc()

Scrittura di un carattere su file
int r = fputc(char c)
int r = putc(char c)
int r = fputchar()
int r = putchar()

Lettura di una stringa da file
char *p = fgets(char *buf, int len, FILE *fp)
char *p = gets(char *buf)

Scrittura di una stringa su file
char *p = fputs(char *s, FILE *fp)
char *p = puts(char *s)

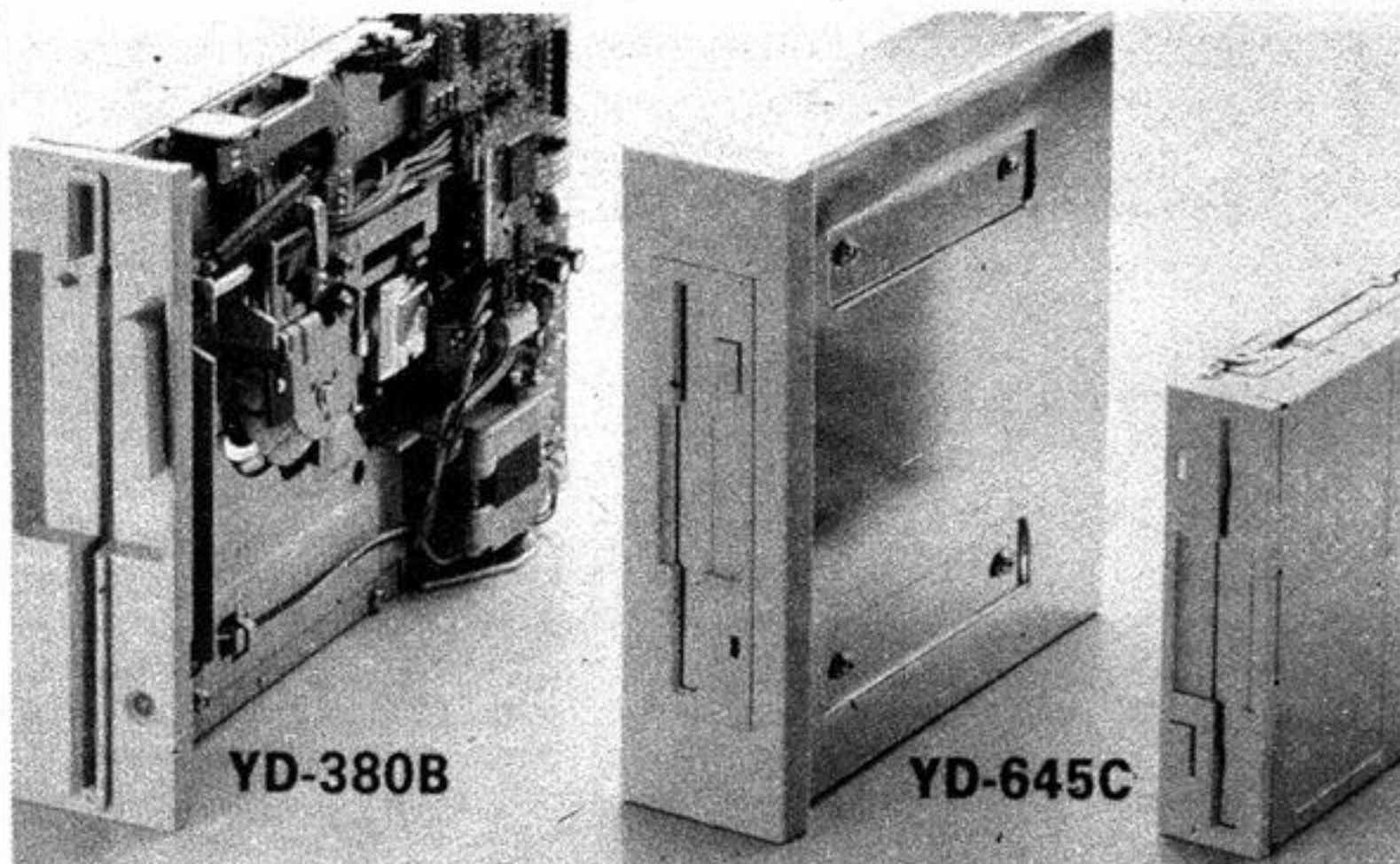
Fig. 2 Funzioni supplementari di lettura e scrittura a Liv.2

con read() (dal file indicato dall'handle fh restituito da open()) cinquecento caratteri, memorizzandoli nell'area puntata dal contenuto di "s", che è appunto un puntatore (l'indirizzo di) una sequenza di bytes (caratteri) ampia cinquecento locazioni.

Se il numero di caratteri letti, restituito dalla funzione open(), è inferiore a quanto rischiesto, si segnala l'errore.

Scrittura

Scrivere dati in un file è un'operazione gemella a quella di lettura, in ambedue i livelli. In effetti, come si vede dalla figura 1, la sintassi formale delle istruzioni è identica. Basta semplicemente pensare che invece di trasferire i dati dal file nel buffer, si esegue la procedura inversa. Tutti gli altri significati sono identici, compreso il valore restituito, che indica il numero di bytes effettivamente scritti. Ad esempio:

```
#include <stdio.h>
#include <string.h>
void main( void )
{
    int fh;
    unsigned tot;
    char s[ 500 ];
    strcpy( &s[0], "Ormellese" );
    fh = open( "Monica", O_WRONLY );
    if ( fh == -1 ) exit( 5 );
    tot = write( fh, s, strlen(s) );
    if ( tot != strlen(s) ) {
        printf( "Errata scrittu-
ra!\n" );
    .. .. ..
    close( fh );
}
```

...esegue la scrittura del contenuto della stringa "s", ovvero la stringa *"Ormelle-*

Bibliografia

Uno, o più, articoli di CCC non possono certamente pretendere di insegnare la programmazione in linguaggio C, specialmente a chi non ha nemmeno una infarinatura su di esso o addirittura sulla programmazione. A questi ultimi possiamo suggerire di muovere i primi passi con linguaggi più "semplici" ed adatti allo scopo, come il Basic. Ai secondi possiamo, invece, dare una lista di volumi che illustrano la programmazione in linguaggio C, più o meno specificamente per Amiga.

Vorremmo ribadire il concetto che è quasi indispensabile conoscere la lingua inglese per imparare a programmare a fondo Amiga in C, per almeno due buoni motivi. Innanzitutto la documentazione ufficiale della Commodore Amiga, qui riportata, è tuttora interamente in lingua inglese. In lingua italiana esistono solo pochissimi testi, in generale di qualità modesta, essendo traduzioni più o meno malfatte di testi tecnici inglesi di problematica traduzione.

Inoltre il parco di programmi in C di "pubblico dominio" circolanti per il mondo, sulle banche dati o sui dischetti, è scritto interamente in lingua inglese. Dal momento che è vitale, per un apprendista programmatore, potere studiare listati scritti da altri, ne deriva una ulteriore importanza della necessità di capire quanto meno l'inglese tecnico, per altro non certamente difficile una volta conosciuti i termini più comuni (spesso usati anche in lingua italiana).

Ecco dunque l'elenco dei testi consigliati *(titolo, autore, editrice, codice)*:

The C Programming Language. Kernighan e Ritchie. Prentice Hall. ISBN 0-13-110163-3. (Disponibile anche in lingua italiana dalla Jackson).

Inside the Amiga with C. John Berry. Howard Sams and Company. ISBN 0-672-22625-1.

Amiga C for Beginners. Schaun. Abacus/Data Becker. ISBN 1-55755- 045-X.

Amiga C for Advanced Programmers. Bleek, Schulz. Abacus/Data BeckerISBN 1-55755-046-8.

Amiga ROM Kernel Reference Manual: Libraries and Devices (Revised and updated). Commodore Amiga Inc. Addison Wesley. ISBN 0-201- 18187-8.

Amiga ROM Kernel Manual Reference Manual: Includes and Autodocs (Revised and updated). Commodore Amiga Inc. Addison Wesley. ISBN 0-201-18177-0.

Amiga Hardware References Manual (Revised and Updated). Commodore Amiga Inc. Addison Wesley. ISBN 0-201-18157-6.

Il Manuale dell'AmigaDOS. Commodore Amiga Inc. IHT Editoriale. ISBN 88-7803-002-3.

Mapping the Amiga. Anderson, Thompson. Compute! Books. ISBN 0- 87455-195-1.

The Kickstart guide to the Amiga. Parkinson, Bolley. Ariadne Software. ISBN 0-95129-210-2.

Linguaggio C per Amiga. Callegari, Feletto. GR Edizioni. ISBN 88- 85201-00-8.

se" come ricopiata tramite l'apposita funzione di libreria **strcpy()** precedentemente, nel file chiamato *"Monica"* aperto in sola lettura con open(). Nel caso il numero di bytes scritti non sia eguale alla lunghezza della stringa; calcolata con strlen(), si genera un messaggio di errore.

Chiusura di un file

La chiusura di un file avviene con le funzioni **fclose()** e **close()**, a cui si passa un handle di file di livello appropriato.

Solitamente si osserva, nei programmi, che si trascura completamente il valore restituito da queste funzioni. In effetti sarebbe buona regola testare che il valore restituito non indichi un errore, nel qual caso il sistema vorrebbe indicare che non è riuscito felicemente a scrivere l'ultimo blocco o sequenza di bytes nel file. Tali dati, quindi, potrebbero essere andati persi senza che il nostro programma si accorga di nulla e possa provvedere. La circostanza di mancata chiusura di file è effettivamente rara in un sistema come AmigaDOS, comunque se si stanno sviluppando programmi di una certa importanza, è bene tenere conto anche di questa pericolosa possibilità di disfunzionamento.

Altre funzioni

Vi sono altre funzioni di base per la lettura e scrittura di file, che accedono a semplici caratteri o stringhe, a livello 2. L'elenco è riportato in figura 2.

Le funzioni **fputc()** e **putc()** sono identiche, scrivendo il primo parametro nel file il cui handle è passato come secondo parametro. Anche le funzioni **fputchar()** e **putchar()** sono identiche tra loro, ma inviano il carattere verso **stdout**, ovvero verso lo schermo video (per default) oppure il device o file indicato sulla linea CLI di chiamata del programma con l'operatore di redirezione "**>**". Le funzioni restituiscono il carattere EOF (definito nel file di inclusione "**stdio.h**") in caso di errore, ed il carattere scritto in caso positivo.

Analogamente le funzioni **fgetc()** e **getc()** leggono e restituiscono un carattere dal file indicato dal filehandle di argomento, oppure letto da **stdin** (tipicamente la tastiera, oppure il device logico od il file indicato sulla linea CLI di chiamata del programma con l'operatore "**<**") nel caso di fgetchar() e getchar(). Il carattere è EOF se si è incontrato un errore o la fine del file.

Le funzioni di stringa operano in maniera analoga, trasferendo una sequenza di caratteri chiusa da un byte di NULL secondo la convenzione del C, letta o scritta sul file di cui si passa l'handle come argomento.

Si noti che queste funzioni utilizzano un buffer di lettura e scrittura gestito automaticamente dalle funzioni di supporto inserite dal linker del compilatore. Ciò significa che anche scrivendo o leggendo un carattere per volta, il sistema userà comunque scrivere o leggere in un'area di memoria, ad evitare lenti e continui accessi al disco, se possibile. Ritorneremo prossimamente a parlare di queste funzioni di gestione dello I/O.

Un programma di esempio

Riportiamo, a titolo esemplificativo, un programma completo che svolge una semplice funzione: contare il numero di caratteri presenti in un file ASCII e presentarne quindi la statistica. Esso illustra abbastanza bene un semplice programma di gestione dei files, con varie funzioni di supporto a livello 2.

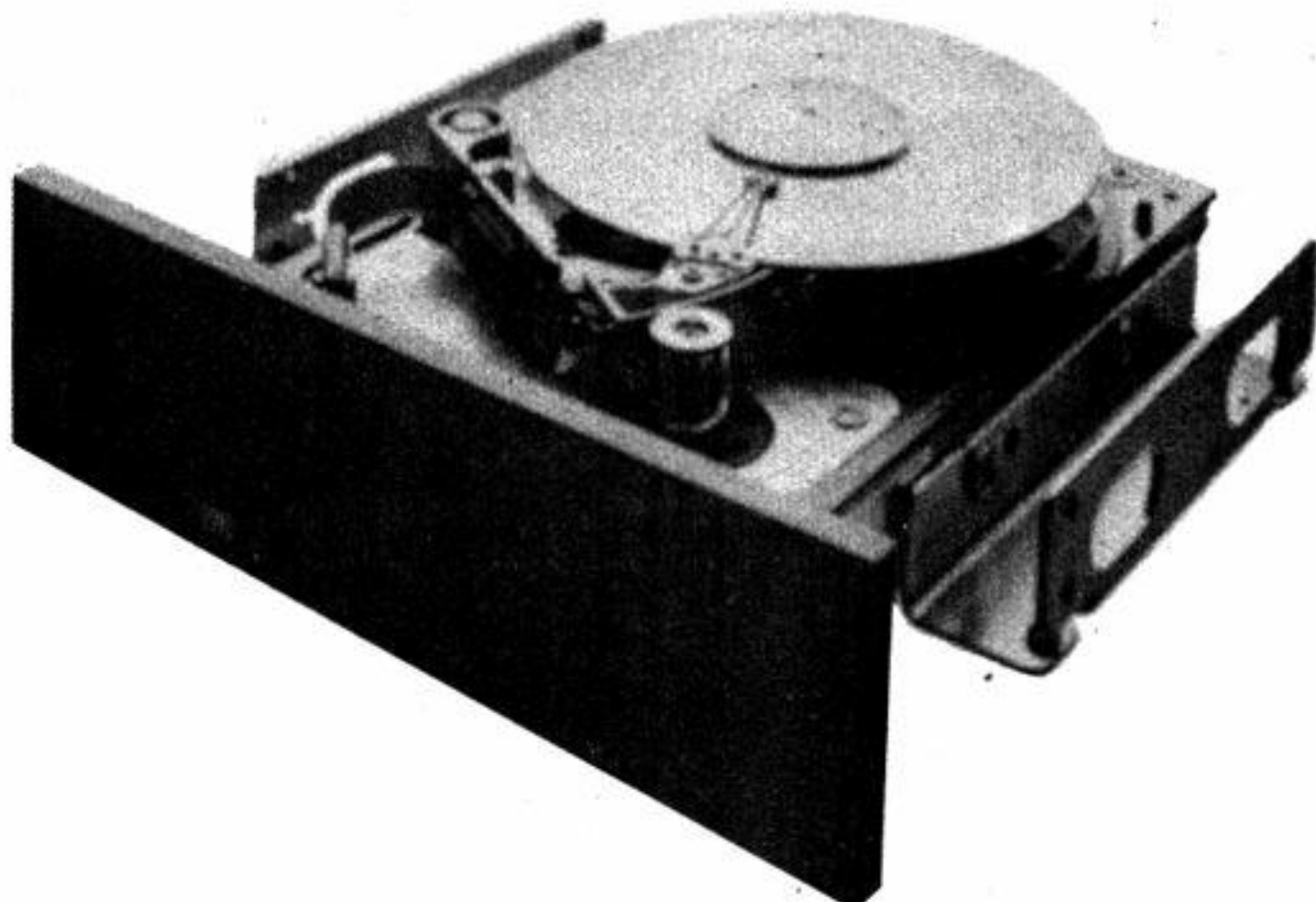
Il listato è stato numerato per consentirne un commento migliore: ovviamente i numeri di linea **non devono essere inseriti** nel listato sorgente!

Si noti che alla linea **21** si usa la funzione **scanf()** per leggere il nome del file da aprire, richiesto con un output su schermo (**stdout**) generato con **printf()**. Lo handle viene memorizzato alla linea **23** nell'omonima variabile. Se l'apertura non è riuscita, ovvero l'handle vale NULL, il programma chiama la funzione **die()** generando quindi un messaggio di avvertimento ed uscendo con **exit()**, nella funzione **die()** stessa.

Altrimenti, inizia alla linea **29** un ciclo con **while()** che legge ripetutamente il file aperto sinchè non si incontra un carattere di fine file (EOF). Il carattere letto viene convertito in **maiuscolo** alla linea **31** con la funzione **toupper()** e, se si rivela alfanumerico (come verifica la funzione **isalpha()**), il suo valore numerico viene utilizzato direttamente per indicare un indice di matrice da incrementare.

Ricordiamo che in C sono perfettamente identiche le seguenti due linee per incrementare una variabile:

```
var = var + num
var += num
```

...che abbiamo usato per incrementare automaticamente la variabile contatrice. Questa, **count**, è stata definita all'esterno della funzione principale per un semplice motivo: così facendo il compilatore provvede automaticamente ad azzerarla **prima** dell'esecuzione, mentre le variabili schiera o matrici dinamiche allocate all'interno delle funzioni hanno valori indefiniti.

La Redazione, per potenziare la propria rete di collaboratori,

CERCA ESPERTI

in possesso dei seguenti requisiti:

1- Possesso di almeno uno dei due computer che saranno oggetto di trattazioni su queste pagine (Ms-Dos compatibili / Amiga)
2- Conoscenza reale e approfondita di almeno due linguaggi di programmazione moderni (Turbo Pascal, Turbo C, Quick Basic, Quick C, Assembly 68XXX, Assembly 80X86).
3- Conoscenza reale e approfondita dell'uso di almeno un Word Processor per Ms - Dos e/o Amiga (è gradita la capacità di usare pacchetti D.T.P.).
4- Reale capacità di sviluppare programmi e articoli attenendosi strettamente alle indicazioni fornite dalla Redazione.

I candidati ideali sono residenti nell'Hinterland milanese; desiderano svolgere una collaborazione esterna, in completa autonomia; sono in grado di portare a termine articoli e programmi nei tempi stabiliti. Gli interessati possono contattare ***telefonicamente*** *la Redazione, allo scopo di fissare un appuntamento per l'esame dettagliato delle proposte redazionali e delle richieste degli aspiranti collaboratori.*

Per informazioni, tel: 02 / 57.60.63.10

di N. F. Spoto

PER AMIGA, ECCO UNO SPRITE IN "C"

Chi ha già sperimentato gli sprite in Amigabasic sicuramente vorrà provare a realizzarli anche in C; magari con un pizzico di Assembler

Tutti noi abbiamo visto, in almeno un programma per Amiga, quelle figure animate che sono a dir poco indispensabili per un videogioco degno di tal nome.

E che dire, del resto, del semplice puntatore del mouse? Tutti sappiamo che tale oggetto è anch'esso uno sprite, cioè un "fantasma"; per gestirne uno, i più intraprendenti userebbero certamente le istruzioni **Object** di **AmigaBasic** senza sapere che è possibile ottenerli anche usando il linguaggio **C**.

Si digiti il programma di queste pagine, lo si **compili** e **linki** con il nome di **Sprite**; se tutto è andato bene, possiamo farlo partire chiamandolo per nome dalla finestra CLI (oppure **Shell**), digitando semplicemente Sprite e premendo il tasto Return. Dopo una domanda relativa al numero dello sprite che si intende gestire (e che può esser compreso fra **0** e **7**, anche se lo 0 è già occupato dal puntatore del mouse e non può essere usato), apparirà una finestra, mentre il mouse si porterà dietro uno strano compagno a forma di palloncino.

Si potrebbe credere che sia stato ridefinito il puntatore del mouse, ma ciò non è vero: si sposti la finestra e si vedrà chiaramente che il palloncino è realmente un'unità autonoma dal mouse.

BITPLANE 0	BITPLANE 1	
0	0	trasparente
0	1	registro colore 1
1	0	registro colore 2
1	1	registro colore 3

Tabella dei bitplane di Amiga

Un po' di teoria

Su Amiga uno **sprite** è un'immagine sovrapposta al video **larga 16 pixels** e lunga anche quanto l'intero schermo. Per definire uno sprite abbiamo quindi bisogno di due elementi: la sua **lunghezza** (la **larghezza** è sempre 16 pixels) e i **dati** che ne definiscono la forma. Tali dati sono rappresentati da una sequenza di **Words**, lunghe **sedici bit**. Per ogni linea dello sprite, usando solo quattro colori, sono necessarie due words; l'intero sprite sarà quindi formato da due gruppi di words, chiamati in gergo piani di bit, **bit-**

planes o bitmaps. I colori si determinano in funzione dei due piani di bit, come visibile in tabella.

Nel programma pubblicato, verso la fine, sono presenti alcune linee in **Assembler** che definiscono i bitplanes; sono stati usati dati a 32 bits, ovvero delle longs (**dc.l**, nel programma), in modo che ogni long contenga una word del bitplane 0 ed una del bitplane 1.

Ciò semplifica la vita in quanto, all'interno della macchina, le words devono alternarsi in modo tale che si succedano una della bitmap 0 ed una della bitmap 1. All'inizio e alla fine dei dati devono essere presenti due words, poste a zero, che servono per usi interni del computer.

Il programma in C, e qui passiamo alla pratica, può fare riferimento a tali dati dichiarandoli array di words esterne senza segno (**extern UWORD bitmap[ ]**); in questo modo, come si dovrebbe sapere, bitmap verrà considerato come un puntatore ad UWORD.

Prima di definire lo sprite, il programma apre una finestra sul video. Anche se ciò è il primo giochetto che si impara a fare con Intuition, non è del tutto inutile ricordare come ciò avviene. Dobbiamo, prima di tutto, definire una struttura **NewWindow**, cioè un insieme di dati che, racchiusi sotto un unico nome (il nome della struttura, appunto), permettono di determinare le caratteristiche della finestra: posizione, altezza, larghezza, ecc. Ciò è possibile con la dichiarazione...

```
struct NewWindow Finestra={...};
```

... dove NewWindow è definita nel file include **'intuition/intuition.h'**.

Per aprire la finestra abbiamo bisogno di una funzione che si trova nella libreria di Intuition: si tratta di **OpenWindow( )**, che apre la finestra e restituisce un valore nullo in caso di errore. Inutile dire che il lettore deve sincerarsi dell'esistenza di tale libreria sul dischetto di sistema, pena l'emissione di un messaggio di errore. Per usare la funzione dobbiamo aprire la libreria di Intuition: **OpenLibrary( )** fa al caso nostro. Anch'essa restituisce zero in caso di errore.

Ma ritorniamo ai nostri sprites. I dati definiti (larghezza e forma) vanno inseriti in una struttura definita nel file include **'graphics/sprite.h'** e chiamata, nel listato, **SimpleSprite**; eccone la definizione completa:

```
struct SimpleSprite{
   UWORD *posctldata;
   UWORD height;
   UWORD x,y;
   UWORD num;
};
```

Come si vede, non c'è un posto specifico per il nostro array bitmap, ma ne riparleremo fra poco. Per ordinare all'Amiga la disponibilità di uno sprite, usiamo la funzione **GetSprite( )**. Essa richiede due parametri: un **puntatore** alla struttura SimpleSprite ed il **numero** di sprite richiesto, sempre compreso fra 0 e 7. Se tutto è a posto possiamo comunicare al computer che i dati per la forma dello sprite si trovano all'indirizzo puntato da **inizio**, che è la copia in Chip Ram di bitmap; i dati per lo sprite devono sempre trovarsi, infatti, in chip ram. Allo scopo provvede la funzione **ChangeSprite( )**, che richiede l'indicazione di vari elementi: anzitutto un puntatore alla ViewPort (non è qui il caso di discutere che cosa sia una ViewPort, basta sapere che possiamo ottenere un puntatore ad essa tramite la funzione **ViewPortAddress( )**); un puntatore alla struttura SimpleSprite interessata, nel nostro caso chiamata **sprite**, e un puntatore alla bitmap. La funzione depositerà i dati di inizio nel membro **posctldata** della struttura SimpleSprite e visualizzerà lo sprite sul video.

Gestione dei colori

Proviamo adesso a modificare i colori dello sprite. Ogni sprite ha a disposizione quattro registri colore, ma solo gli ultimi tre vengono utilizzati; in effetti la combinazione di bitplanes 0-0 rende il pixel trasparente.

Dei 32 registri colore dell'Amiga, gli ultimi sedici sono destinati agli otto sprites, secondo la tabella:

```
reg. colore 16...19 spr. 0 e 1
reg. colore 20...23 spr. 2 e 3
reg. colore 24...27 spr. 4 e 5
reg. colore 28...31 spr. 6 e 7
```

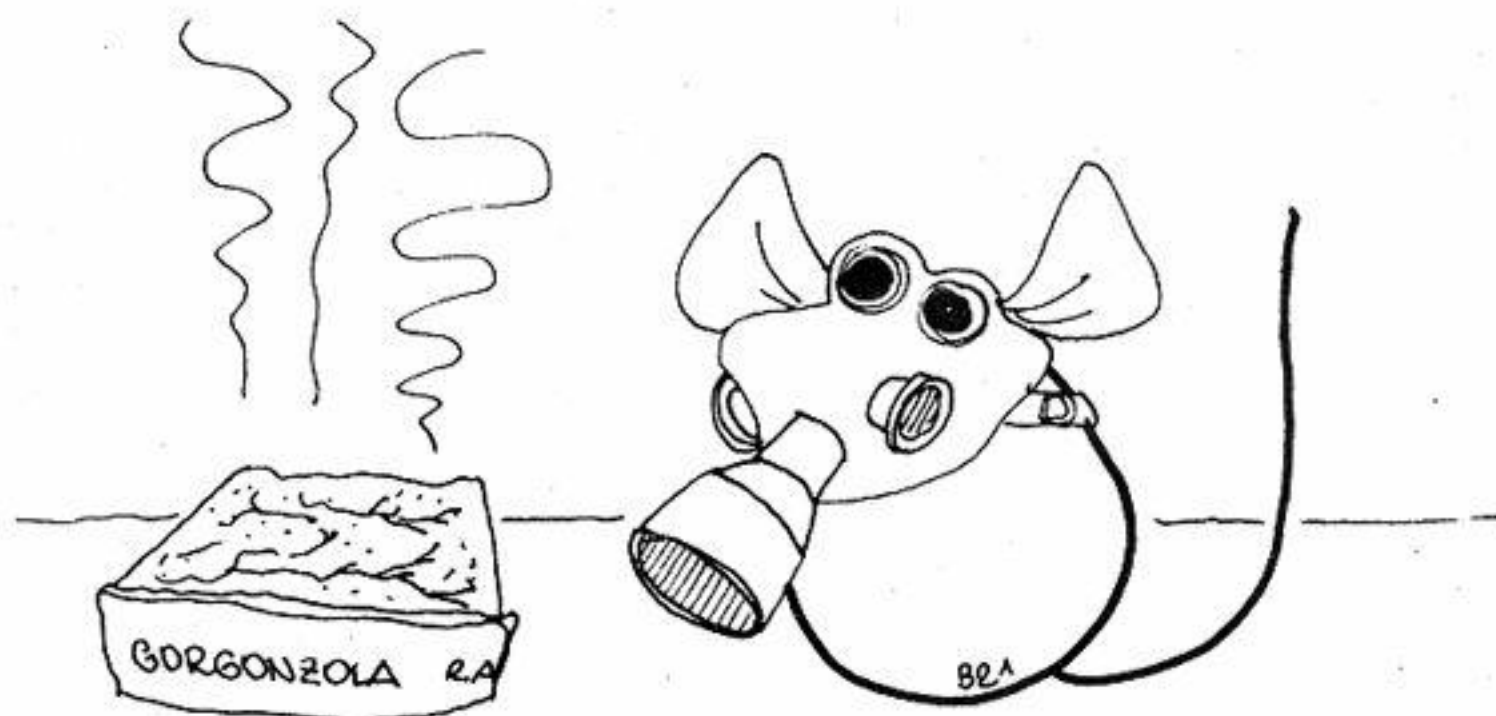
Ciò significa che gli sprites condividono i colori a gruppi di due. Tenendo conto che lo sprite 0 è il puntatore del mouse, non si potranno quindi mai cambiare i colori dello sprite 1 senza cambiare quelli del puntatore del mouse.

Per cambiare i registri dei colori ci serviamo della funzione...

SetRGB4(vp,regclr,R,G,B)

...dove **vp** è il solito puntatore alla ViewPort, **regclr** è il registro colore (espresso come long) ed **R**, **G**, **B** sono tre longs che contengono le tre componenti colore, **rosso**, **verde** e **blu**, espresse con valori compresi fra **0** e **15**.

A questo punto non ci manca che far seguire allo sprite i movimenti del mouse. Dopo aver aggiornato la struttura SimpleSprite, ci si ricordi che bisogna dare un **ChangeSprite( )**, altrimenti il cambiamento non avrà effetto. In questo caso non dobbiamo cambiare la forma dello sprite, ma solo la sua posizione; la forma sarà la solita, cioè quella puntata da posctldata. Se l'utente chiude la finestra, non ci resta che chiudere bottega; ci si ricordi solo della funzione **FreeSprite( )**.

Speriamo che la passeggiata tra bit e words sia stata utile. Qualcuno potrà pensare che non è stata agevole, ma abbiamo dimenticato che cosa era spesso necessario fare sui computer ad otto bit?

```
/* sprite.c: per Amiga by N.F. Spoto
Per compilare con l'AZTEC: cc sprite.c (RETURN) ln sprite.o -lc (RETURN) */

#include <intuition/intuition.h>
#include <exec/types.h>
#include <exec/memory.h>
#include <graphics/sprite.h>
#define ALTSPR 16              /* altezza dello sprite in pixels */
#define SPAZIO 2L*(2*ALTSPR+4) /* numero di bytes che occupa lo sprite */
struct Library          *IntuitionBase, *GfxBase, *OpenLibrary();
struct Window           *finpunt, *OpenWindow();
struct IntuiMessage     *msg, *GetMsg();
struct ViewPort         *vp, *ViewPortAddress();
struct SimpleSprite     sprite;
struct NewWindow        finestra={
  10,10,                   /* coordinate dell'angolo in alto a sinistra */
  300,30,                  /* larghezza ed altezza */
  -1, -1,                  /* colori dei dettagli e dell'interno standard */
  CLOSEWINDOW,             /* IDCMP flag */
  WINDOWCLOSE|WINDOWDRAG,  /* caratt. finestra: chiusura e spostabilita' */
  0,0,                     /* gadgets e checkmark assenti */
  (UBYTE *)"Dimostrativo sprite",       /* titolo */
  0,0,          /* puntatore allo schermo e bitmap aggiuntive non usati */
  -1,-1,                   /* dimensioni minime e massime della finestra */
  -1,-1,                   /* (non usate: la finestra non e' modificabile)*/
  WBENCHSCREEN             /* tipo */
};
long            numspr, GetSprite();
extern UWORD    bitmap[];
void            *AllocMem();
void main()
{
  char buff[256];
  puts("numero dello sprite da utilizzare ?");
  gets(buff);
  numspr=atoi(buff); /* atoi e' come la funzione BASIC 'VAL' */
  ApriTutto();
  DefSprite();
  MuoviSprite();
  Fine("Ciao!");
}
ApriTutto()
{
  IntuitionBase=OpenLibrary("intuition.library",0);
  if (!IntuitionBase) Fine("Non riesco ad aprire intuition.library");
  GfxBase=OpenLibrary("graphics.library",0);
  if (!GfxBase) Fine("Non riesco ad aprire graphics.library");
  finpunt=OpenWindow(&finestra);
  if (!finpunt) Fine("Non riesco ad aprire la finestra");
  vp=ViewPortAddress(finpunt); /* in vp l'indirizzo della ViewPort */
}
DefSprite()
{
  int i;
  long regclr;
  UWORD *inizio;
  inizio=AllocMem(SPAZIO,MEMF_CHIP);      /* richiedo lo spazio per inizio */
```

```
  for (i=0;i<SPAZIO/2;i++) inizio[i]=bitmap[i];   /* copio bitmap in inizio*/
  sprite.x=0;            /* determino i valori di partenza per lo sprite:   */
  sprite.y=0;            /* modificare una struttura SimpleSprite non ha    */
  sprite.height=ALTSPR; /* un effetto diretto:bisogna dare uno ChangeSprite*/
  if(GetSprite(&sprite,numspr)<0)  /* lo sprite numspr adesso e' questo */
    Fine("GetSprite fallito");
  ChangeSprite(vp,&sprite,inizio); /* copia in sprite i dati di bitmap */
  SetRGB4(vp,(regclr=16+(numspr/2)*4)+1,15L,0L,0L); /*determino i 3 colori*/
  SetRGB4(vp,regclr+2,0L,15L,0L);                   /* per lo sprite: */
  SetRGB4(vp,regclr+3,15L,15L,15L);                 /* rosso, verde, bianco*/
}
MuoviSprite()
{
  int finaperta=TRUE;
  while (finaperta){
    sprite.x=finpunt->MouseX; /* sposta lo sprite seguendo le coordinate  */
    sprite.y=finpunt->MouseY; /* del mouse ricavabili dalla struct Window */
    ChangeSprite(vp,&sprite,sprite.posctldata); /* aggiorno lo sprite */
if ((msg=GetMsg(finpunt->UserPort))->Class==CLOSEWINDOW)
  finaperta=FALSE; /* termina se si e' premuto il gadget di CLOSEWINDOW*/
  }
}
Fine(stringa)
char *stringa;
{
  printf("\n%s\n\n",stringa); /* stampa il messaggio di abbandono e libera:*/
  if (sprite.num) FreeSprite((long)sprite.num);   /* ... lo sprite */
  if (finpunt) CloseWindow(finpunt);              /* ... la finesatr */
  if (GfxBase) CloseLibrary(GfxBase);             /* ... la libr. intuition*/
  if (IntuitionBase) CloseLibrary(IntuitionBase); /* ... e grafica */
  exit(0);
}
/* Dati per la forma dello sprite */
#asm
_bitmap:
  dc.l %00000000000000000000000000000000
  dc.l %00000011110000000110000111000011
  dc.l %00001111111100001000110000110001
  dc.l %00011111111110000001100000011000
  dc.l %00111111111111000011000000001100
  dc.l %01111111111111100110000000000110
  dc.l %01111111111111100110000000000110
  dc.l %11111111111111111000000000000011
  dc.l %11111111111111111000000000000011
  dc.l %11111111111111111000000000000011
  dc.l %11111111111111111000000000000011
  dc.l %01111111111111100110000000000110
  dc.l %01111111111111100110000000000110
  dc.l %00111111111111000011000000001100
  dc.l %00011111111110000001100000011000
  dc.l %00001111111100001000110000110001
  dc.l %00000011110000000110000111000011
  dc.l %00000000000000000000000000000000
#endasm
```

di Valentino Spataro

UN CATALOGO PER AMIGA

Amigados: è un "linguaggio" le cui potenzialità sono ancora sconosciute alla maggior parte dei suoi utenti

Amiga, come è noto, "conosce" solo il linguaggio macchina, anche se se la cava sufficientemente bene con il **Basic** o con Amiga **Dos**.

Con Amiga, più che con il C/64, queste considerazioni diventano vitali.

Se, infatti, il nostro problema consiste nel muovere una decina di sprite su e giù per lo schermo per una presentazione, un movimento fluido, veloce e continuo non sarà mai ottenibile con un programma scritto in Amigabasic; forse compilandolo (a patto di usare un buon compilatore) si può ottenere una discreta (ma non ottimale) resa.

Programmando in **C**, oppure in **Assembler**, il problema si risolve facilmente(!), ma il rovescio della medaglia consiste, appunto, nella difficoltà di imparare un nuovo linguaggio e nella grande quantità di tempo necessario per sviluppare il programma.

Se, invece, vogliamo creare un semplice archivio in cui memorizzare le nostre audiocassette (o un breve programma che controlli il codice fiscale), un qualsiasi interprete si proporrà come soluzione ideale.

Se, poi, il nostro problema consiste nell'effettuare frequenti copie di sicurezza, un **File Batch** contenente una successione di comandi **Copy** permetterà di effettuare le copie desiderate, utilizzando un solo comando ed eliminando, per di più, il rischio di digitare comandi errati.

Se qualcuno, pertanto, afferma che l'unico linguaggio di programmazione degno di nota è il l'Assembler oppure il C (o qualcun altro ancora) diffidate! La scelta dipende unicamente dalla quantità di tempo a nostra disposizione per stendere il programma e dal tipo di problema che ci accingiamo a risolvere; tutto il resto è un di più.

Programmi in Amigados

Per trascrivere il **file batch** (programma scritto nel linguaggio Amiga Dos) pubblicato in queste pagine, potete servirvi di qualsiasi programma di videoscrittura, purchè registri i files in formato **ASCII** standard, tenendo conto che anche gli spazi hanno un significato preciso.

Salvatelo con il nome **Cataloga**, caricate **CLI** (oppure **SHELL**, a vostro piacere) e digitate, ad esempio:

```
EXECUTE RAM:CATALOGA
```

...se il file si trova in **Ram:** (assicuratevi, però, che lo abbiate registrato anche su un dischetto: non si mai...).

Come funziona

Cominciamo con un po' di sana e utilissima teoria. Prima di tutto bisogna ricordare che un file di comandi in Amigados è un **file di testo** e non una **sequenza di codici** che il computer è in grado di interpretare. Amiga ha bisogno, ogni volta che incontra un comando, di sapere il "luogo" in cui questo è memorizzato, dal momento che tutti i comandi del Dos sono comandi **esterni** (cioè devono esser caricati da disco per essere eseguiti). Tale informazione viene ricercata nei percorsi indicati dal comando **PATH**, oltre che (automaticamente) anche nella directory "**C**" del disco di avviamento, luogo ove si presume siano presenti tutti i comandi.

Bisogna ora tener presente che gli utenti di Amiga possono non avere due drive. Nasce così un problema: Amiga, nel caso disponga di un solo drive, si serve del drive **DF0:** per caricare ed eseguire i vari comandi. Ciò implica la lettura del contenuto del disco presente nel drive DF0: che risulta, però, essere il disco di avviamento.

Per risolvere il problema (evitando di effettuare infiniti scambi dei due dischetti altrimenti necessari, cioè Workbench e quelli di archivio) basta servirsi degli altri drive di cui dispone Amiga, cioè la **RAM:** (gestibile facilmente anche dai principianti) oppure la **RAD:** (per gli esperti che sanno usarla).

La soluzione cui ci riferiremo è la RAM: perchè non richiede alcuna operazione supplementare; si ricorda infatti che per creare una RAM: basta inviare un qualsiasi comando ad essa (anche se non "esiste" ancora) perchè il computer provveda autonomamente a crearla (basta un CD RAM: impartito da CLI per crearla).

In pratica, il programma occupa la memoria in questo modo:

1) i comandi vengono trasferiti nella directory C: della RAM:; quest'ultima viene creata, se non esiste (operazione necessaria perchè il comando **COPY** della versione **1.2** del WB non provvede da solo

```
cd sys:c
assign rc: ram:c
if not exists rc:
 makedir rc:
endif
path rc:
copy copy rc:
copy ask rc:
copy if rc:
copy skip rc:
copy endif rc:
copy echo rc:
copy dir rc:
copy cd rc:
copy rename rc:
copy join rc:
copy delete rc:
copy sort rc:
copy else rc:
copy lab rc:

echo >ram:pippo1 "elenco dischi"
lab domanda
 ask "vuoi catalogare un disco in df0: (y/n) ?"
 if not warn
  skip fine
 endif
 echo "sto lavorando"
 cd df0:
 cd >ram:pippo4
 dir >ram:pippo2 df0: all
  rename ram:pippo1 ram:pippo3
 join ram:pippo3 ram:pippo4 ram:pippo2 as ram:pippo1
 delete ram:pippo3
 skip domanda back
lab fine
 delete ram:pippo2
 delete ram:pippo4
ask "vuoi uscire senza registrare (y/n) ?"
  if warn
   quit
  endif
ask "vuoi formattare un disco in df0: (y/n) ?"
  if warn
   sys:system/format drive df0: name elencodischi
  endif
 ask "inserire disco contente elenco in df0: e pre-
mere return"

echo "il prg provvede a fare una copia di sicurezza"
 echo "dell'elenco precedente"
 echo >df0:mom "elenco dischi"
 if exists df0:mom
  delete df0:mom
 endif
 if exists df0:elencodisco.new
  copy df0:elencodisco.new df0:elencodisco.old
  rename df0:elencodisco.new as df0:mom
 else
  echo >df0:mom "elenco dischi"
 endif
 join df0:mom ram:pippo1 as df0:elencodisco.new
lab out
 delete ram:pippo1
 delete df0:mom
 ask "vuoi un altro file con i dati ordinati ?"
 if warn
   echo "Metto in ordine"
   sort df0:elencodisco.new to df0:elencodisco.ord
 endif
 echo "operazioni completate"
quit
```

a creare un directory);

2) forza il computer a cercare i comandi all'interno di RAM:C (tramite il comando PATH:RC, dove RC: è RAM:C per il comando **Assign**);

3) mette nel file provvisorio **PIPPO4** il nome del disco attuale; mentre mette in **PIPPO2** il suo contenuto; cambia il nome del file contenente l'elenco (chiamato **PIPPO1**) in **PIPPO3**; unisce nel nuovo file PIPPO1 i files PIPPO3, PIPPO4 e PIPPO2 nell'ordine indicato, in modo tale che l'ultimo disco archiviato si metta in fondo alla lista e con il nome prima dell'elenco dei files.

Premesso che alla prima lettura avrete fatto una certa(!) fatica per capire il procedimento decisamente cervellotico, con qualche ulteriore lettura, a seconda delle capacità maniacali ricorsive, capirete che è questo il cuore del programma; tutto il resto è un contorno per farlo funzionare.

Come usare il programma

Una volta copiato e registrato il programma, vi consigliamo di farne una copia in RAM: e farlo partire da lì. In ogni caso, prima di lanciarlo, copiatelo in RAM:. Per i novellini riportiamo qui di seguito le operazioni da fare:

1) copiare con un word processor o un text editor il programma pubblicato;

2) salvarlo **due** volte: la prima volta con il nome: **DF0:CATALOGA** e la seconda con **RAM:CATALOGA**

3) aprire una finestra di CLI oppure SHELL;

4) scrivere: **RAM:CATALOGA** e aspettare una decina di secondi.

5) rispondere alle domande.

Le domande che compariranno saranno di intuitiva risposta in quanto il comando **ASK** di Amigados permette solo una risposta affermativa (**Y**) o negativa (**N** oppure tasto Return "a vuoto").

Per il resto, il programma è facilmente comprensibile anche per chi conosce poco l'Amigados ma un po' di inglese; e non è certamente poco.

Modificando

Purtroppo il programma ha un limite, rappresentato dal formato dell'istruzione **DIR ALL**, che è l'unica che permette la visualizzazione di tutti i files presenti in tutte le subdirectories.

Il comando visualizza, in ogni linea del video, due nomi di files; inoltre, nel caso di subdirectory, non evidenzia in quale subdirectory è presente il file visualizzato. Comunque, se i catalogatori sul mercato non vi soddisfano perchè troppo lenti o troppo affezionati a Guru malefici, il file batch di queste pagine, usato in coppia con un valido word processor, si presenta come valida alternativa.

QUANTO COSTA IL TUO COMMODORE

Amiga 2000 - L. 2.715.000

Microprocessore Motorola MC68000 - Clock 7.16MHz - Kickstart ROM - Memoria RAM: 1 MByte - 3 chip custom per DMA, Video, Audio, I/O - 5 Slot di Espansione Amiga Bus 100 pin Autoconfig™ - 1 Slot di Espansione 86 pin per Schede Coprocessore - 2 Slot di Espansione compatibili AT/XT - 2 Slot di Espansione compatibili XT - 2 Slot di Espansione Video - 1 Floppy Disk Drive da 3 1/2", 880 KBytes - Porta seriale RS232C - Sistema Operativo single-user, multitasking AmigaDOS - Compatibilità MS-DOS XT/AT disponibile con schede interne Janus (A2088 - A2286) - Monitor escluso

Amiga 500 - L. 995.000

Microprocessore Motorola MC68000 - Clock 7.16 MHz - Kickstart ROM - Memoria RAM: 512 KBytes - 3 Chip custom per DMA, Video, Audio, I/O - 1 Floppy Disk Driver da 3 1/2", 880 KBytes - Porta seriale RS232C - Porta parallela Centronics

Videomaster 2995 - L. 1.200.000

Desk Top Video - Sistema per elaborazini video semiprofessionale composto da genlock, digitalizzatore e alloggiamento per 3 drive A2010 - Ingressi videocomposito (2), RGB - Uscite Videocomposito, RF, RGB + sync -

Floppy Disk Driver A 1010 - L. 335.000

Floppy Disk Driver - Drive esterno da 3 1/2" - Capacità 880 KBytes - Collegabile a tutti i modelli della linea Amiga, alla scheda A2088 e al PC1

Floppy Disk Drive A 2010 - L. 280.000

Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" - Capacità 880 KBytes - Collegabile ad Amiga 2000

Hard Disk A 590 - L. 1.750.000

Hard Disk+Controller+RAM - Scheda Controller - Hard Disk da 3 1/2" 20 MBytes - 2 MBytes "fast" RAM - Collegabile all'Amiga 500

Scheda Janus A 2088 + A 2020 - L. 1.050.000

Scheda Janus XT+Floppy Disk Drive da 5 1/4", 360 KBytes - Scheda Bridgeboard per compatibilità MS-DOS (XT) in Amiga 2000 - Microprocessore Intel 8088 - Coprocessore matematico opzionale Intel 8087

A2286+A2020 - L. 1.985.000

Scheda Janus AT+Floppy Disk Drive da 5 1/4", 1.2 MBytes - Scheda Bridgeboard per compatibilità MS-DOS (AT) in Amiga 2000 - Microprocessore Intel 80287 - Clock 8 MHz - RAM: 1 MBytes on-board - Floppy Disk Controller on-board - Floppy Disk Driver disegnato per l'installazione all'interno dell'Amiga 2000 -

Scheda A2620 - L. 2.700.000

Scheda Processore Alternativo 32 bit - Scheda per 68020 e Unix - Microprocessore Motorola MC68020 - Coprocessore matematico Motorola MC68881 (opzionale MC68882)

Scheda A Unix - L. 3.250.000

Sistema Operativo AT&T Unix System V Release 3 - Per Amiga 2000 con scheda A2620 e Hard Disk 100 MBytes

Hard Disk A2092+PC5060 - L. 1.020.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+2092 - L. 1.240.000

Hard Disk e controller - Hard Disk 3 1/2" ST506 - Capacità formattata 20 MBytes

Hard Disk A2090+A2094 - L. 1.900.000

Stesse caratteristiche del kit A2092 ma con disco da 40 MBytes

Espansione di memoria A2058 - L. 1.149.000

Espansione di memoria - Scheda di espansione per Amiga 2000 - Fornita con 2 MBytes "fast" RAM, espandibile a 4 o 8 MBytes

Scheda Video A2060 - L. 165.000

Modulatore video - Scheda modulatore video interna per Amiga 2000 - Uscite colore e monocromatica - Si inserisce nello slot video dell'Amiga 2000

Genlock Card A2301 - L. 420.000

Genlock - Scheda Genlock semiprofessionale per Amiga 2000 - Permette di miscelare immagini provenienti da una sorgente esterna con immagini provenienti dal computer

Professional Video Adapter Card A2351 - L. 1.500.000

Professional Video Adapter - Scheda Video Professionale per Amiga 2000 (B) - Genlock qualità Broadcast - Frame Grabber - Digitalizzatore - Include software di controllo per la gestione interattiva
(Disponibile da maggio '89)

A501 - L. 300.000

Espansione di memoria - Cartuccia di espansione di memoria da 512 KBytes per A500

A520 - L. 45.000

Modulatore RF - Modulatore esterno A500 - Permette di connettere qualsiasi televisore B/N o colori ad Amiga 500

A Scart - L. 27.000
Cavo di collegamento A500/A2000 con connettore per televisione SCART

Monitor a colori 1084 - L. 595.000
Monitor a colori ad alta risoluzione - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor a colori 2080 - L. 770.000
Monitor a colori ad alta risoluzione e lunga persistenza - Tubo 14" Black Matrix antiriflesso - Pitch 0.39 mm - Frequenza di raster 50 Hz - Compatibile con Amiga 500/2000, PC (tutta la gamma), C64 e C128

Monitor Monocromatico A2024 - L. 1.235.000
Monitor monocromatico a fosfori "bianco-carta" - Turbo 14" antiriflesso - (Disponibile da marzo '89)

PC60/40 - L. 7.812.000
Microprocessore Intel 80386 - Coprocessore matematico opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 funzioni - Sistema Operativo MS-DOS 3 2.1 - Interprete GW-Basic

PC60/40C - L. 8.127.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 60/80 - L. 10.450.000
Microprocessore Intel 80386 - Coprocessore opzionale Intel 80387 - Clock 8 o 16 MHz selezionabile via software e da tastiera - Memoria RAM: 2.5 MBytes - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Floppy Disk Drive opzionale da 3 1/2", 1.44 MBytes - 1 Hard Disk da 80 MBytes - 2 Porte parallele Centronics - Mouse video EGA (compatibile MDA - Hercules - CGA). Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Ambiente Operativo Microsoft Windows/386 - Interprete GW-Basic

PC60/80C - L. 10.700.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC40/20 - L. 4.100.000
Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/20C - L. 4.350.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC 40/40 - L. 5.285.000
Microprocessore Intel 80286 - Coprocessore matematico opzionale Intel 80287 - Clock 6 o 10 MHz selezionabile via software, hardware o da tastiera - Memoria RAM: 1 MByte - 1 Floppy Disk Drive da 5 1/4", 1.2 MBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232 - Porta parallela Centronics - Scheda video AGA multistandard (MDA - Hercules - CGA) Emulazioni disponibili via hardware e software - Monitor monocromatico 14" - Tastiera avanzata 102 tasti con 12 tasti funzione - Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC40/40C - L. 5.535.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

1352 - L. 78.000
Mouse - Collegabile con Microsoft Bus Mouse - Collegabile direttamente a PC1, PC10/20 - III, PC40 - III

PC910 - L. 355.000
Floppy Disk Drive - Drive interno aggiuntivo da 3 1/2" per PC10/20-I-II-III - Capacità 360 o 720 KBytes selezionabile tramite "config. sys" - Corredo di telaio di supporto per l'installazione in un alloggiamento per un drive da 5 1/4" - Interfaccia identica ai modelli da 5 1/4"

PC1 - L. 995.000
Microprocessore Intel 8088 - 1 Floppy Disk Drive da 5 1/4" - Porta seriale RS232C - Porta parallela Centronics - Monitor monocromatico 12" - Tastiera 84 tasti - Sistema Operativo MS-DOS 3.2 - Interprete GW-Basic

PCEXP1 - L. 640.000
PC Expansion Box - Box esterno di espansione per PC 1 - Alimentatore aggiuntivo incluso - Contiene 3 Slot di Espansione compatibili Ibm XT - Alloggiamento per Hard Disk da 5 1/4" - Si posiziona sotto il corpo del PC1 e viene collegato tramite degli appositi connettori

PC10-III - L. 1.360.000
Microprocessore Intel 8088 Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - Memoria RAM: 640 KBytes - 2 Floppy Disk Drive da 5 1/4", 360 KBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC10-IIIC - L. 1.675.000
Stessa configurazione ma con monitor 14" a colori mod. 1084

PC20-III - L. 2.095.000
Microprocessore Intel 8088 - Clock 4.77 MHz 9.54 MHz (double) selezionabile via software e da tastiera - 1/4", 360 KBytes - 1 Hard Disk da 20 MBytes - Porta seriale RS232C - Porta parallela Centronics - Porta Mouse per Mouse Commodore 1352 (compatibile Microsoft Bus Mouse) - Tastiera avanzata 102 con 12 tasti funzione Sistema Operativo MS-DOS 3.21 - Interprete GW-Basic

PC20-IIIC - L. 2.410.000

Stessa configurazione ma con monitor 14" a colori mod. 1084

Nuovo C64 - L. 325.000

Nuovo Personal Computer CPU 64 KBytes RAM - Vastissima biblioteca software disponibile - Porta seriale Commodore - Porta registratore per cassette - Porta parallela programmabile -

C128D - L. 895.000

Personal Computer CPU 128 KBytes RAM espandibile a 512 KBytes - ROM 48 KBytes - Basic 7.0 - Tastiera separata - Funzionante in modo 128,64 o CP/M 3.0 - Include floppy disk drive da 340 KBytes

Floppy Disk Drive 1541 II - L. 365.000

Floppy Disk Drive - Floppy Disk Drive da 5 1/4" singola faccia - Capacità 170 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

Floppy Disk Dirve 1581 - L. 420.000

Floppy Disk Drive da 3 1/2" doppia faccia - Capacità 800 KBytes - Alimentazione separata - Compatibile con C64, C128, C128D

1530 - L. 55.000

Registratore a cassette per C64, C128, C128D

Accessori per C64 - 128D

1700 - Espansione di memoria - Cartuccia di espansione di memoria a 128 KBytes per C128 **- L. 170.000**

1750 - Espansione di memoria - Cartuccia di espansione di memoria 512 KBytes per C128 - **L. 245.000**

1764 - Espansione di memoria - Cartuccia di espansione di memoria a 256 KBytes per C64 - Fornita di alimentatore surdimensionato - **L. 198.000**

16499 - Adattatore Telematico Omologato - Collegabile al C64 - Permette il collegamento a Videotel, P.G.E. e banche dati **L. 149.000**

1399 - Joystick - Joystick a microswitch con autofire - **L. 29.000**

1351 - Mouse - Mouse per C64, C128, C128D - **L. 72.000**

Monitor Monocromatico 1402 - L. 280.000

Monitor monocromatico a fosfori "bianco-carta" - Turbo 12" antiriflesso - Ingresso TTL - Compatibile con tutta la gamma PC

Monitor Monocromatico 1404 - L. 365.000

Monitor monocromatico a fosfori ambra - Turbo 14" antiriflesso a schermo piatto - Ingresso TTL - Compatibile con tutta la gamma PC - Base orientabile

Monitor Monocromatico 1450 - L. 470.000

Monitor monocromatico BI-SYNC a fosfori "bianco-carta" - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Monitor a colori 1802 - L. 445.000

Monitor a colori - Turbo 14" - Collegabile a C64, C128, C128D

Monitor monocromatico 1900 - L. 199.000

Monitor monocromatico a fosfori verdi - Turbo 12" antiriflesso - Ingresso videocomposito - Compatibile con tutta la gamma Commodore

Monitor a colori 1950 - L. 1.280.000

Monitor a colori BI-SYNC alta risoluzione - Turbo 14" antiriflesso - Ingresso analogico e digitale - Doppia frequenza di sincronismo orizzontale per compatibilità con adattatori video MDA, Hercules, CGA, EGA e VGA

Stampante MPS 1230 - L. 465.000

Stampante a matrice di punti - Testina a 9 aghi - 120 cps - Bidirezionale - 80 colonne - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

MPS 1230R - L. 19.000

Nastro per stampante

Stampante MPS 1500C - L. 495.000

Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia parallela Centronics - Compatibile con la gamma Amiga e PC

MPS1500R - L. 37.000

Nastro a colori per stampante

MPS1500R - L.37.000

Nastro a colori per stampante

Stampante MPS 1550C - L. 575.000

Stampante a colori a matrice di punti - Testina a 9 aghi - 130 cps - Bidirezionale - 80 colonne - Supporta nastro a colori o nero - Near Letter Quality - Stampa grafica - Fogli singoli e modulo continuo - Trascinamento a trattore e/o frizione - Interfaccia seriale Commodore e parallela Centronics - Compatibile con tutti i prodotti Commodore

I COMMODORE POINT

LOMBARDIA

Milano
• AL RISPARMIO - V.LE MONZA 204
• BCS - VIA MONTEGANI 11
• BRAHA A. - VIA PIER CAPPONI 5
• E.D.S. - C.SO PORTA TICINESE 4
• FAREF - VIA A. VOLTA 21
• FLOPPERIA - V.LE MONTENERO 31
• GBC - VIA CANTONI 7 - VIA PETRELLA 6
• GIGLIONI - V.LE LUIGI STURZO 45
• L'UFFICIO 2000 - VIA RIPAMONTI 213
• LOGITEK - VIA GOLGI 60
• LU - MEN - VIA SANTA MONICA 3
• MARCUCCI - VIA F.LLI BRONZETTI 37
• MELCHIONI - VIA P. COLLETTA 37
• MESSAGGERIE MUSICALI - GALLERIA DEL CORSO 2
• NEWEL - VIA MAC MAHON 75
• PANCOMMERZ ITALIA - VIA PADOVA 1
• SUPERGAMES - VIA VITRUVIO 38
• 68000 E DINTORNI - VIA WASHINGTON 91
Provincia di Milano
• GINO FERRARI CENTRO HI-FI - VIA MADRE CABRINI 44 - S. ANG. LODIGIANO
• F.LLI GALIMBERTI - VIA NAZIONALE DEI GIOVI 28/36 - BARLASSINA
• TECNOLUX - VIA PIETRO NENNI 5 - BERNATE TICINO
• OGGIONI & C. - VIA DANTE CESANA 27 - CARATE BRIANZA
• AL RISPARMIO - VIA U. GIORDANO 57 - CINISELLO BALSAMO
• GBC - V.LE MATTEOTTI 66 - CINISELLO BALSAMO
• CASA DELLA MUSICA - VIA INDIPENDENZA 21 - COLOGNO MONZESE
• PENATI - VIA VERDI 28/30 - CORBETTA
• EPM SYSTEM - V.LE ITALIA 12 - CORSICO
• P.G. OSTELLARI - VIA MILANO 300 - DESIO
• CENTRO COMPUTER PANDOLFI - VIA CORRIDONI 18 - LEGNANO
• COMPUTEAM - VIA VECELLIO 41 - LISSONE
• M.B.M. - C.SO ROMA 112 - LODI
• L'AMICO DEL COMPUTER - VIA CASTELLINI 27 - MELEGNANO
• BIT 84 - VIA ITALIA 4 - MONZA
• IL CURSORE - VIA CAMPO DEI FIORI 35 - NOVATE MIL.
• I.C.O. - VIA DEI TIGLI 14 - OPERA
• R & C ELGRA - VIA SAN MARTINO 13 - PALAZZOLO MIL.
• ESSEGIEMME SISTEMI SAS - VIA DE AMICIS 24 - RHO
• TECNO. - CENTRO - VIA BARACCA 2 - SEREGNO
• NIWA HARD&SOFT - VIA B. BUOZZI 94 - SESTO SAN GIOV.
• COMPUTER SHOP - VIA CONFALONIERI 35 - VILLASANTA
• ACTE - VIA B. CREMIGNANI 13 - VIMERCATE
• IL COMPUTER SERVICE SHOP - VIA PADANA SUPERIORE 197 - VIMODRONE
Bergamo
• D.R.B. - VIA BORGO PALAZZO 65
• TINTORI ENRICO &C. - VIA BROSETA 1
• VIDEO IMMAGINE - VIA CARDUCCI c/o CITTA' DI MERCATO
Provincia di Bergamo
BERTULEZZI GIOVANNI - VIA FANTONI 48 - ALZANO LOMBARDO
• COMPUTER SHOP - VIA VITTORIO VENETO 9 - CAPRIATE SAN GERVASIO
• B M R - VIA BUTTARO 4/T - DALMINE
• MEGABYTE 2 - VIA ROMA 61/A - GRUMELLO
• OTTICO OPTOMETRISTA ROVETTA - P.ZZA GARIBALDI 6 - LOVERE
• COMPUTER POINT - VIA LANTIERI 52 - SARNICO
• A.B. INFORMATICA - STRADA STATALE CREMASCA 66 - URGNANO
Brescia
• MASTER INFORMATICA - VIA F.LLI UGONI 10/B
PROVINCIA DI BRESCIA
• MISTER BIT - VIA MAZZINI 70 - BRENO
• CAVALLI PIETRO - VIA 10 GIORNATE 14 BIS - CASTREZZATO
• VIETTI GIUSEPPE - VIA MILANO 1/B - CHIARI
• MEGABYTE - P.ZZA MALUEZZI 14 - DESENZANO DEL GARDA
• BARESI RINO &C. - VIA XX SETTEMBRE 7 - GHEDI
• INFO CAM - VIA PROVINCIALE 3 - GRATACASOLO
• "PAC-LAND" di GARDONI - CENTRO COM.le - LA CASA DI MARGHERITA D'ESTE - VIA GIORGIONI 21
Como
• IL COMPUTER - VIA INDIPENDENZA 90
• 2M ELETTRONICA - VIA SACCO 3
Provincia di Como
• ELTRON - VIA IV NOVEMBRE 1 - BARZANO'
• DATA FOUND - VIA A. VOLTA 4 - ERBA
• CIMA ELETTRONICA - VIA L. DA VINCI 7 - LECCO
• FUMAGALLI - VIA CAIROLI 48 - LECCO
• RIGHI ELETTRONICA - VIA G. LEOPARDI 26 - OLGIATE COMASCO
Cremona
• MONDO COMPUTER - VIA GIUSEPPINA 11/B
• PRISMA - VIA BUOSO DA DOVARA 8
• TELCO - P.ZZA MARCONI 2/A
Provincia di Cremona
• ELCOM - VIA IV NOVEMBRE 56/58 - CREMA
• EUROELETTRONICA - VIA XX SETTEMBRE 92/A - CREMA
Mantova
• COMPUTER CANOSSA - GAL. FERRI 7
• 32 BIT - VIA C. BATTISTI 14
• ELET. di BASSO - V.LE RISORGIMENTO 69
Provincia di Mantova
• CLICK - ON COMPUTER - S.S. GOITESE 168 - GOITO
Pavia
• POLIWARE - C.SO C. ALBERTO 76
• SENNA GIANFRANCO - VIA CALCHI 5
Provincia di Pavia
• A. FERRARI - C.SO CAVOUR 57 - MORTARA
• LOGICA MAINT - V.LE M.TE GRAPPA 32 - VIGEVANO
• M. VISENTIN - C.SO V. EMANUELE 76 - VIGEVANO
Sondrio
• CIPOLLA MAURO - VIA TREMOGGE 25
Provincia di Sondrio
• FOTONOVA - VIA VALERIANA 1 - S.PIETRO DI BERBENNO
Varese
• ELLE - EFFE - VIA GOLDONI 35
• IL C.TRO ELET. - VIA MORAZZONE 2
• SUPERGAMES - VIA CARROBBIO 13
Provincia di Varese
• BUSTO BIT - VIA GAVINANA 17-BUSTO A.
• MASTER PIX - VIA S.MICHELE 3 - BUSTO A.
• PUNTO UFFICIO - VIA R.SANZIO 8 - GALLARATE
• GRANDI MAGAZZINI BOSSI - VIA CLERICI 196 - GERENZANO
• J.A.C. - C.so MATTEOTTI 38 - SESTO C.

PIEMONTE

Alessandria
• BIT MICRO - VIA MAZZINI 102
• SERV. INFOR. - VIA ALESSANDRO III 47
Provincia di Alessandria
• SONY ITALIANA - VIA G. MANARA 7 - CASALE MONFERRATO
• SGE ELETTRONICA - VIA BANDELLO 19 - TORTONA
• COMPUTER TEMPLE - VIA F. CAVALLOTTI 13 - VALENZA
Asti
• ASTI GAMES - C.SO ALFIERI 26
• RECORD - C.SO ALFIERI 166/3 (Galleria Argenta)
Cuneo
• ROSSI COMPUTERS - C.SO NIZZA 42
Provincia di Cuneo
• PUNTO BIT - C.SO LANGHE 26/C - ALBA
• BOSETTI - VIA ROMA 149 - FOSSANO
• COMPUTERLAND - VIA MAZZINI 30/32 - SALUZZO
Novara
• PROGRAMMA 3 - V.LE BUONARROTI 8
• PUNTO VIDEO - C.so RISORGIMENTO 39/B
Provincia di Novara
• COMPUTER - VIA MONTE ZEDA 4 - ARONA
• ALL COMPUTER - C.SO GARIBALDI 106 - BORGOMANERO
• S.P.A. - C.SO DISSEGNA 21/BIS - DOMODOSSOLA
• ELLIOTT COMPUTER SHOP - VIA DON MINZONI 32 - INTRA
• TRISCONI VALERIA - VIA MAZZINI 90 - OMEGNA
Torino
• ABA ELETTRONICA - VIA C. FOSSATI 5/P
• ALEX COMPUTER E GIOCHI - C.SO FRANCIA 333/4
• COMPUTER HOME - VIA SAN DONATO 46/D
• COMPUTING NEW - VIA M. POLO 40/E
• C.D.M. ELETTR. - VIA MAROCHETTI 17
• DE BUG - C.SO V. EMANUELE II 22
• DESME UNIVERSAL - VIA S.SECONDO 95
• FDS ALTERIO - VIA BORGARO 86/D
• IL COMPUTER - VIA N. FABRIZI 126
• MICRONTEL - C.SO D. degli ABRUZZI 28
• PLAY GAMES SHOP - VIA C. ALBERTO 39/E
• RADIO TV MIRAFIORI - C.SO UNIONE SOVIETICA 381
• SMT ELETTRONICA - VIA BIBIANA 83/bis
Provincia di Torino
• PAUL E CHICO VIDEOSOUND - VIA V.EMANUELE 52 - CHIERI
• BIT INFORMATICA - VIA V. EMANUELE 154 - CIRIE'
• HI - FI CLUB - C.SO FRANCIA 92C - COLLEGNO
• MISTER PERSONAL - VIA CATTANEO 52 - FAVRIA
• I.C.S. - VIA TORINO 73 - IVREA
• DAG - VIA I MAGGIO 40 - LUSERNA S. GIOVANNI
• EUREX - C.SO INDIPENDENZA 5 - RIVAROLO CANAVESE
• DIAM INFORMATICA - C.SO FRANCIA 146 bis - RIVOLI
• FULLINFORMATICA - VIA V.VENETO 25 - RIVOLI
• GAMMA COMPUTER - VIA CAVOUR 3A-3B - SET.TORINESE
Vercelli
• ELETTROGAMMA - C.SO BORMIDA 27 ang. V.Montanara
• ELETTRONICA - STRADA TORINO 15
Provincia di Vercelli
• C.S.I. TEOREMA - VIA LOSANA 9 - BIELLA
• SIGEST - VIA BERTODANO 8 - BIELLA
• REMONDINO FRANCO - VIA ROMA 5 - BORGOSESIA
• FOTOSTUDIO TREVISAN - VIA XXV APRILE 24/B - COSSATO
• STUDIO FOTOGRAFICO IMARISIO - P.ZZA M. LIBERTA' 7 - TRINO

VENETO

Belluno
• UP TO DATE - VIA V. VENETO 43
Provincia di Belluno
• GUERRA COMPUTERS - V.LE MAZZINI 10/A - FELTRE
Padova
• BIT SHOP - VIA CAIROLI 11
• COMPUMANIA - VIA T. CAMPOSANPIERO 37
• D.P.R. DE PRATO R. - V.LO LOMBARDO 4
• G.F. MARCATO - VIA MADONNA DELLA SALUTE 51/53
• SARTO COMPUTER - VIA ARMISTIZIO 79
Provincia di Padova
• COMPUTER SERVICE - BORGO TREVISO 150 - CITTADELLA
Treviso
• BIT 2000 - VIA BRANDOLINI D'ADDA 14
• GUERRA EGIDIO & C. - V.LE CAIROLI 95
Provincia di Treviso
• DE MARIN COMPUTERS - VIA MATTEOTTI 142 - CONEGLIANO
• SIDESTREET - VIA SALVO D'ACQUISTO 8 - MONTEBELLUNA
• FALCON ELETTROAUDIOVIDEO - VIA TERRAGGIO 116 - PREGANZIOL
Venezia
• GUERRA EGIDIO & C. - VIA BISSUOLA 20/A - MESTRE
• TELERADIO FUGA - SAN MARCO 3457
Provincia di Venezia
• GUERRA EGIDIO & C. - VIA VIZZOTTO 29 - SAN DONA' DI PIAVE
• REBEL - VIA F. CRISPI 10 - SAN DONA' DI PIAVE
Verona
• CASA DELLA RADIO - VIA CAIROLI 10
• TELESAT - VIA VASCO DE GAMA 8
Provincia di Verona
• UBER - CP 0363(RAG.SOC. DERTA) - VIA MASCAGNI 31 - CASTE' D'AZZANO
• FERRARIN - VIA DEI MASSARI 10 - LEGNAGO
• COMPUTERS CENTER - VIA CANTORE 26 - VILLAFRANCA
Vicenza
• ELET. BISELLO - V.LE TRIESTE 427/429
• SCALCHI MARKET - VIA C.A' BALBI 139
Provincia di Vicenza
• SCHIAVOTTO - VIA ZANELLA 21 - CAVAZZALE
• GUERRA E. & C. - V.LE DELLE INDUSTRIE - MONTECCHIO MAGGIORE

FRIULI VENEZIA GIULIA

Gorizia
• E.CO. ELETTRONICA - VIA F.LLI COSSAR 23
Trieste
• AVANZO GIACOMO - P.ZZA CAVANA 7
• COMPUTER SHOP - VIA P. RETI 6
• COMPUTIGI - VIA XX SETTEMBRE 51
• CTI - VIA PASCOLI 4
Udine
• MOFERT 2 - VIA LEOPARDI 21
• R.T. SISTEM UDINE - VIA L. DA VINCI 99
Provincia di Udine
• IL PUNTO ELETTRONICO - VIA VENDRAMIN 184 - LATISANA
• IDRENO MATTIUSSI &C. - VIA LICINIANA 58 - MARTIGNACCO

TRENTINO ALTO ADIGE

Bolzano
• COMPUTER POINT - VIA ROMA 82/A
• MATTEUCCI PRESTIGE - VIA MUSEO 54
Provincia di Bolzano
• RADIO MAIR-ELECTRO - VIA CENTRALE 70 - BRUNICO
• ELECTRO RADIO HENDRICH - VIA DELLE CORSE 106 - MERANO
• ERICH KONTSCHIEDER - PORTICI 313 - MERANO
• ELECTRO TAPPEINER - P.ZZA PRINCIPALE 90 - SILANDRO
Trento
• CRONST - VIA G. GALILEI 25
Provincia di Trento

• AL RISPARMIO - C.SO VERONA 138 - ROVERETO

LIGURIA
Genova
• ABM COMPUTER - P.ZZA DE FERRARI 24 rosso
• CAPRIOTTI G. - IA MAMIANI 4r - SAMPIERDARENA
• C.tro ELET. - VIA CHIARAVAGNA 10 R - VIA SESTRI 69R
• COM.le SOTTORIPA - VIA SOTTORIPA 115/117
• FOTOMONDIAL - VIA DEL CAMPO 3-5-9-11-13 r
• LA NASCENTE - VIA SAN LUCA 4/1
• PLAY TIME - VIA GRAMSCI 3/5/7 rosso
• RAPPR-EL - VIA BORGORATTI 23 R
Imperia
• CASTELLINO - VIA BELGRANO 44
Provincia di Imperia
• CENTRO HI-FI VIDEO - VIA DELLA REPUBBLICA 38 -SANREMO
• CASTELLINO - VIA GENOVA 48 - VENTIMIGLIA
La Spezia
• I.L. ELETTRONICA - VIA V. VENETO 123
Provincia di La Spezia
• I.L. ELETTRONICA - VIA AURELIA 299 - FORNOLA DI VEZZANO
Savona
• CASTELLINO - C.SO TARDY E BENECH 101
Provincia di Savona
• CELESIA ENZA - VIA GARIBALDI 144 - LOANO

EMILIA
Bologna
• EUROELETTRICA - VIA RANZANI 13/2
• MINNELLA ALTA FEDELTA' - VIA MAZZINI 146/2
• MORINI & FEDERICI - VIA MARCONI 28/C
• STERLINO - VIA MURRI 73/75
Provincia di Bologna
• S.C. COMPUTERS - VIA E. FERMI 4 - CASTEL SAN PIETRO
• S.P.E. INFORMATICA - VIA DI MEZZO PONENTE 385 - CREVALCORE
• ARCHIMEDE SISTEMI - VIA EMILIA 124 - S. LAZZARO DI SAVENA
Modena
• CO - EL - VIA CESARI 7
• ORSA MAGGIORE - P.ZZA MATTEOTTI 20
• VIDEO VAL WILLY COMPUTERS - VIA CANALETTO 223
Provincia di Modena
• NEW MEDIA SYSTEM - VIA ROMA 281 - SOLIERA
Parma
• BABARELLI G. - VIA B. PARENTE 14/A/B
Provincia di Parma
• PONGOLINI - VIA CAVOUR 32 - FIDENZA
Piacenza
• COMPUTER LINE - VIA G. CARDUCCI 4
• DELTA COMPUTER - VIA M. DELLA RESISTENZA 15/G

TEGGIO EMILIA
• COMPUTERLINE - VIA SAN ROCCO 10/C
• POOL SHOP - VIA EMILIA S. STEFANO 9/C
Provincia di Reggio Emilia
• MACCHIONI - VIA STATALE 467 - CASALGRANDE
ROMAGNA
Ferrara
• BUSINESS POINT - VIA CARLO MAYER 85
Forlì
• COMPUTER VIDEO CENTER - VIA CAMPO DI MARTE 122
Provincia di Forlì
• TOP BIT - VIA VENETO 12 - FORLIMPOPOLI
• COMPUTER HOUSE - V.LE TRIPOLI 193/D - RIMINI
• EASY COMPUTER - VIA LAGOMAGGIO 50 - RIMINI

REPUBBLICA S. MARINO
Ravenna
• COMPUTER HOUSE - VIA TRIESTE 134
Provincia di Ravenna
• ARGNANI - P.ZZA DELLA LIBERTA 5/A - FAENZA
• ELECTRON INFORMATICA - VIA F.LLI CORTESI 17 - LUGO
• P.L.Z. INFORMATICA - P.ZZA SERCOGNANI 6 - FAENZA

TOSCANA
Arezzo
• DELTA SYSTEM - VIA PIAVE 13
Firenze
• ATEMA - VIA BENEDETTO MARCELLO 1a-1b
• ELETTRONICA CENTOSTELLE - VIA CENTO STELLE 5/a-b
• HELP COMPUTER - VIA DEGLI ARTISTI 15-A
• TELEINFORMATICA TOSCANA -VIA BRONZINO 36
Provincia di Firenze
• WAR GAMES - VIA R. SANZIO 126/A - EMPOLI
• NEW EVM COMPUTER - VIA DEGLI INNOCENTI 2 - FIGLINE VALDARNO
• C.tro INFOR. - VIA ZNOJMO 41 - PONTASSIEVE
• COSCI F.LLI - VIA ROMA 26 - PRATO
• BARBAGLI C. ELET. - VIA F. BONI 80 - PRATO
Grosseto
• COMPUTER SERVICE - VIA DELL'UNIONE 7
Livorno
• ETA BETA - VIA SAN FRANCESCO 30
• FUTURA 2 - VIA CAMBINI 19
Provincia di Livorno
• PUNTO ROSSO - VIA BARONTINI 28 - PIOMBINO
Provincia di Lucca
• IL COMPUTER - V.LE COLOMBO 216 - LIDO DI CAMAIORE
• SANTI VITTORIO - VIA ROMA 23 - S. ROMANO GARFAGNANA
• TOP GAMES - VIA S. ANDREA 122 - VIAREGGIO
Massa
• EURO COMPUTER - P.ZZA G. BERTAGNINI 4
Carrara
• RADIO LUCONI - VIA ROMA 24/B
Pisa
• ELECTRONIC SERVICE - VIA DELLA VECCHIA TRANVIA 10
• PUCCINI S.- CP 1199 (RAG.SOC. MAREX) - VIA C.CAMMEO 64
• TONY HI-FI - VIA CARDUCCI
Provincia di Pisa
• M.C. INFORMATICA - VIA DEL CHIESINO 4 - PONTEDERA (PI)
Pistoia
• ELECTRONIC SHOP - VIA DEGLI SCALZI 3
Provincia di Pistoia
• ZANNI &C. - C.SO ROMA 45 - MONTECATINI T.

Siena
• R. BROGI - P.ZZA GRAMSCI 28
• VIDEO MOVIE - VIA GARIBALDI 17
Provincia di Siena
• ELETTRONICA di BIFOLCHI - VIA DI GRACCIANO NEL CORSO 111 - MONTEPULCIANO

LAZIO
• CENTRO INF. - D.R.R. srl - TEL. 06-5565672

UMBRIA
Perugia
• MIGLIORATI - VIA S. ERCOLANO 3-10
Provincia di Perugia
• COMPUTER STUDIO'S - VIA IV NOVEMBRE 18/A - BASTIA UMBRA
• WARE - VIA DEI CASCERI 31 - CITTA'DI CASTELLO
Terni
• CGS SOFTWARE HOUSE - VIA DONIZETTI 71/A

BASILICATA
Matera
• G. GAUDIANO ELECTRONICS - VIA ROMA ang. XX SETTEMBRE 1

PUGLIA
Bari
• ARTEL - VIA GUIDO D'ORSO 9
• COMPUTER'S ARTS - V.LE MEUCCI 12/B
• PAULICELLI S. & F. - VIA FANELLI 231/C
Provincia di Bari
• F. FAGGELLA - C.SO GARIBALDI 15 - BARLETTA
• G.FAGGELLA - P.ZZA D'ARAGONA 62A - BARLETTA
• LONUZZO G. - VIA NIZZA 21 - CASTELLANA
• TECNOUFF. - VIA RICASOLI 54 - MONOPOLI
• TANGORRA N. - C.SO V.EMANUELE 130/B - TRIGGIANO
Brindisi
• MARANGI E NICCOLI - VIA PROV. SAN VITO 165
Provincia di Brindisi
• MILONE G. - VIA S.F. D'ASSISI 219 - FRANCAVILLA FONTANA
Foggia
• BOTTICELLI G. - VIA SAV POLLICE 2
• E.C.I. COMPUTER - VIA ISONZO 28
• LA TORRE - V.LE MICHELANGELO 185
Provincia di Foggia
• IL DISCOBOLO - VIA T. SOLIS 15 - SAN SEVERO
Lecce
• BIT - VIA 95 REGG.NTO FANTERIA 87/89
Provincia di Lecce
• TECNO UFFICIO - P.ZZA GIOVANNI XXIII 10 - GALLIPOLI
• CEDOK INFORMATICA - VIA UMBERTO I 116 - TRICASE
Taranto
• ELETTROJOLLY C.tro - VIA DE CESARE 13
• TEA - TEC. ELET. AV. - VIA R. ELENA 101

CAMPANIA
Provincia di Avellino
• FLIP FLOP - VIA APPIA 68 - ATRIPALDA
Benevento
• E.CO. INF. - VIA PEPICELLI 21/25
Caserta
• ENTRY POINT - VIA COLOMBO 31
• O.P.C. - VIA G. M. BOSCO 24
Provincia di Caserta
• M.P. COMPUTER - VIA NAPOLI 30 - MADDALONI
• DAMIANO - C.SO V. EMANUELE 23 -ORTA DI ATELLA
• FUSCO B. - VIA NAPOLI 24 - VAIRANO PATERNORA (FRAZ. VAIRANO SCALO)
• LINEA CONTABILE - VIA OSPEDALE 72/76 - SESSA A. (CE)
Napoli
• BABY TOYS - VIA CISTERNA DELL'OLIO 5/BIS
• CASA MUSICALE RUGGIERO - P.ZZA GARIBALDI 74 (INT. STAZ. F.F. S.S.)
• C.tro ELET. CAMPANO - VIA EPOMEO 121
• CI.AN - GALLERIA VANVITELLI 32
• CINE NAPOLI - VIA S. LUCIA 93/95
• DARVIN - CALATA SAN MARCO 26
• GIANCAR 2 - P.ZZA GARIBALDI 37
• ODORINO - L.GO LALA 22 A-B
• R 2 - VIA F. CILEA 285
• SAGMAR - VIA S. LUCIA 140
• TOP VIDEO - TOP COMPUTER - VIA S. ANNA DEI LOMBARDI 12
• VIDEOFOTOMARKET - VIA S. BRIGIDA 19
Provincia di Napoli
• ELECTRONIC DAY - VIA DELLE PUGLIE 17 - CASORIA
• TUFANO - S.S. SANNITICA 87 KM 7 - CASORIA
• SOF SUD - V.LE EUROPA 59 - CASTEL/MARE DI STABIA
• ELETTRONICA 2000 - C.SO DURANTE 40 - FRATTAMAGGIORE
• SPADARO - VIA ROMANI 93 - MADONNA DELL'ARCO
• GATEWAY - VIA NAPOLI 68 - MUGNANO
• VISPINI & DI VUOLO - VIA A.ROSSI 4 - POMPEI
• SPY CASH & CARRY - P.ZZA ARENELLA 6/A - NAPOLI
• NUOVA INFORMATICA SHOP - VIA LIBERTA' 185/191 PORTICI
• BASIC COMPUTER - C.SO GARIBALDI 34 - POZZUOLI
• V.C. - C.SO SECONDIGLIANO 562/B - SECONDIGLIANO
• F. ELETTRONICA - VIA SARNO 102 - STRIANO
• TECNO - VIA V. VENETO 48 - TORRE DEL GRECO
Salerno
• COMPUMARKET - VIA BELVEDERE 35
• COMPUTER MARKET - C.SO VITTORIO EMANUELE 23
Provincia di Salerno
• KING COMPUTER - VIA OLEVANO 56 - BATTIPAGLIA
• DIMER POINT - V.LE AMENDOLA 36 - EBOLI
• IACUZIO F. - VIA MUNICIPIO 14 - MERCATO SAN SEVERINO
• COMPUTER SERVICE - VIA L.DA VINCI 81 - SCAFATI

CALABRIA
Catanzaro
• C. & G. COMPUTER - VIA F. ACRI 28
• PAONE S. & F. - VIA F. ACRI 93/99
Provincia di Catanzaro
• COMPUTER HOUSE - VIA BOLOGNA (L.GO OSPEDALE) - CROTONE
• RIOLO F.LLI - VIA VENEZIA 1/7 - CROTONE
• ING. FUSTO S. - C.SO NICOTERA 99 - LAMEZIA TERME
Cosenza
• MAISON DE L'INFORMATIQUE - VIA PASQUALE ROSSI 34/C
• SIRANGELO COMP. - VIA N. PARISIO 25
Provincia di Cosenza
• HI-FI ALFANO G. - VIA BALDACCHINI 109 - AMANTIA
• ELIGIO ANNICCHIARICO &C. - VIA ROMA 21 - CASTROVILLARI
• ALFA COMPUTER - VIA NAZIONALE 341/A - CORIGLIANO SCALO
REGGIO CALABRIA
• CONTROL SYSTEM - VIA S.F DA PAOLA 49 D
• SYSTEM HOU. - VIA FIUME ang. PALESTINO 1
Provincia di Reggio Calabria
• COMPUTER SHOP - V.LE MATTEOTTI 36/38 - LOCRI
• PICIEFFE - C.SO F. S. ALESSIO 19 - TAURIANOVA
SICILIA
• CENTRO INF. - ITALSOFT SRL - TEL. 0935-696090